Käyttäjävetoinen innovaatio

Viestinnän koulutusohjelma
Digitaalinen viestintä
Opinnäytetyö
5.5.2010

Jani Toivonen

Metropolia

TIIVISTELMÄSIVU

Koulutusohjelma	Suuntautumisvaihtoehto
Viestinnän koulutusohjelma	Verkkoviestintä

Tekijä
Jani Toivonen

Työn nimi
Käyttäjävetoinen innovaatio

Työn ohjaaja/ohjaajat
Matti Rantala

Työn laji	Aika	Numeroidut sivut + liitteiden sivut
Opinnäytetyö	5.5.2010	35 + 39

TIIVISTELMÄ

Tämä toiminnallinen opinnäytetyö käsittelee käyttäjävetoista innovaatiota. Työssä on teososa sekä kirjallinen osa. Teososa koostuu neljästä käyttäjäinnovaatiota toiminnallisesti tutkivasta prototyypistä. Innovaatiokirjallisuuden mukaan ympäristömme muutoksilla on ollut käyttäjäinnovaatioita tukeva ja käyttäjiä voimauttava vaikutus. Tämä väite oli perustana työn tutkimukselle. Tämän työn teososa on rajattu aihepiiriin, jossa yhdistetään tietojenkäsittelyä ja elektroniikkaa.

Aluksi selvitettiin, mitä innovaatio on ja mitä tarvitaan innovaatioiden tekemiseen. Sen jälkeen tutkittiin käyttäjän mahdollisia rooleja innovaatioprosessissa. Elektronisten prototyyppien kehitysprojekteilla testattiin käyttäjän mahdollisuuksia innovaatioiden tekemiseen käytännössä. Projekteilla selvitettiin, mitä käyttäjäinnovaatioita tukevia ja niitä estäviä tekijöitä käyttäjä voi kohdata.

Työssä selvisi, että monet ympäristömme muutokset ovat vaikuttaneet voimauttavasti käyttäjien kykyyn innovoida. Käyttäjäinnovaatiot ovat suuri mahdollisuus, mikäli niitä opitaan tukemaan ja hyödyntämään tehokkaasti. Työn lopuksi ehdotetaan muutamia parannuksia käytettyihin kehitystyökaluihin ja toimintatapoihin.

Teos/Esitys/Produktio
4 Prototyyppiä

Säilytyspaikka
Aralis-kirjastokeskus

Avainsanat
Innovaatio, käyttäjävetoinen, sulautettu järjestelmä

Metropolia

Culture

Degree Programme in	Specialisation
Media	New Media Design

Author
Jani Toivonen

Title
User-Driven Innovation

Tutor(s)
Matti Rantala

Type of Work	Date	Number of pages + appendices
Bachelor´s Thesis	5. May 2010	35 + 39

ABSTRACT

The present thesis focuses on user-driven innovation. The study was conducted in two parts. The first part consists of electronic prototypes and the second part comprises the written thesis. According to leading studies in the field of innovation, users are increasingly able to innovate for themselves. The ongoing shift has empowered users and is changing the innovation environment. The object of this thesis is to find out whether these claims hold water.

These hypothesises were tested through simulated processes of user-driven innovation, the electronic prototypes being the innovations. The factors supporting or hindering the users ability to innovate were studied through the prototype processes. The prototypes consist of electronic compositions, which could be described by the terms of embedded-system, physical-computing or ubiquitous computing.

It was discovered that the ongoing shift has empowered users. Open source software, the trend towards open data, open communities and other fairly new phenomenon occurred. A further discovery involved the possibility for innovation, which has spread wider and more evenly than before. User-driven innovation is a great possibility provided that it can be successfully promoted and utilized.

Work / Performance / Project
4 Prototypes

Place of Storage
Aralis-library

Keywords
Innovation, user-driven, embedded system

SISÄLLYS

1 JOHDANTO 2
2 TYÖSSÄ KÄYTETYT KESKEISET TERMIT 3
3 PROTOTYYPIT JA TUTKIMUS 4
3.1 Prototyypit 5
3.2 Toiminnallinen kehittäminen 6
4 INNOVAATIO 8
4.1 Esimerkki innovaatioiden syntymisestä 9
4.2 Mistä innovaatio muodostuu? 10
4.2.1 Tarvetieto 10
4.2.2 Ratkaisutieto 12
4.3 Tiedon epätasapaino ja tahmeus 12
5 EDULLISEN INNOVAATIOTILAN LUOMINEN 13
5.1 Tiedon lähteet ja tahmean tiedon siirtäminen 14
5.2 Työkaluja käyttäjäinnovaatioon ja räätälöityyn tuotesuunnitteluun. 15
5.3 Tiedon avoin jakaminen ja diffuusio 18
5.4 Havaintoja tiedonlähteistä 20
6 MUUTTUVA INNOVAATIOYMPÄRISTÖ 21
6.1 Käyttäjän muuttuva rooli 23
6.2 Avoin Elektroniikka 24
6.3 Avoin tieto 26
6.4 Nopean muutoksen vaikutus 27
7 YHTEENVETO 27
7.1 Käyttäjäinnovaatioita tukevat muutokset 28
7.2 Parannusehdotuksia 29
7.3 Kuudennen sukupolven innovaatio? 31
LÄHTEET 32
LIITTEET

1 JOHDANTO

Tämä työ käsittelee käyttäjävetoista innovaatiota. Käyttäjävetoisen innovaation merkitys on selkeästi ymmärretty ja aihe on saanut paljon huomiota osakseen viime vuosina. Eric Von Hippel on alan pioneeri, ja hänen väittämänsä mukaan ympäristömme muutoksilla on ollut käyttäjäinnovaatioita tukeva ja käyttäjiä voimauttava vaikutus. Tämän väitteen kirjoittamisesta on nyt (2010) neljä vuotta. Työssä tutkin, miten tämä väite pitää paikkansa tänä päivänä, miten käyttäjäinnovaatiot voivat syntyä ja mitkä tekijät mahdollistavat ne.

Työssä on teososa, jossa tutkin näitä aiheita toiminnallisen kehittämisen kautta. Innovaatioita tehdään kaikilla elämänaloilla, mutta tämän työn teososa on rajattu aihepiiriin jossa yhdistetään tietojenkäsittelyä ja elektroniikkaa. Valitsin tämän alueen, koska sen kehitys on ollut nopeaa ja mahdolliset muutokset olisivat siten nopeasti havaittavissa.

Aluksi selvitän, mitä innovaatio on ja mitä tarvitaan innovaatioiden tekemiseen. Tutkin käyttäjän mahdollisuuksia tehdä innovaatioita, rakentamalla elektronisia prototyyppejä kehitystyökalujen avulla. Kuvailen työn aikana havaintojani näistä projekteista, suhteessa olemassa olevaan tietoon käyttäjävetoisesta innovaatiosta. Tarkastelen mitä resursseja käyttäjälle on tarjolla ja mitä haasteita kehitystyö tarjosi. Tekemäni prototyypit ovat elektroniikkaa ja tietojenkäsittelyä yhdisteleviä projekteja, joiden tavoitteena on toimiva mallinnus kuvitteellisesta tuotteesta tai tarpeesta. Prototyyppejä voisi kuvailla termeillä jokapaikan tietotekniikka tai sulautettu järjestelmä. Avaan termien merkityksiä seuraavassa luvussa, mutta kaikessa yksinkertaisuudessaan

tekemäni prototyypit ovat tietojenkäsittelyä ja fyysistä maailmaa yhdistäviä kokeiluja. Työni lopuksi esittelen tutkimuksen tulokset sekä teen yhteenvedon työni käsittelemistä aiheista.

2 TYÖSSÄ KÄYTETYT KESKEISET TERMIT

Innovaatiokirjallisuudessa käytetään runsaasti termistöä, jonka ymmärtäminen on oleellista kokonaisuuden hahmottamiseksi. Tässä luvussa avaan tärkeimpiä työssä käytettyjä termejä. Termistön uutuus ja vakiintuneiden suomenkielisten vastineiden puuttuminen olivat hetkittäin haasteellinen yhdistelmä. Kirjallisuudessa samoja termejä käytetään myös eri asiayhteyksissä ja eri tarkoituksissa, riippuen kirjoittajasta. Niinpä on oleellista työn ymmärtämisen kannalta kuvata, mitä käytetyt termit tarkoittavat tässä työssä.

Jokapaikan tietotekniikka (engl. *ubiquitous computing/pervasive computing*)	Käsite tarkoittaa mikroprosessorien sijoittamista arjen ympäristöön, esineisiin, tiloihin ja myös eläviin olentoihin. Termi *Pervasive Computing* tarkoittaa periaatteessa samaa. ”(Haglund 2006, 32.) Jokapaikan tietotekniikka on huomaamattomasti toimivaa ja ympäristöönsä sulautuvaa, kaikkialla olevaa tietotekniikkaa. Se ei häiritse käyttäjäänsä eikä keskeytä hänen muuta toimintaansa. Se toimii ihmisten ja yritysten arkitoimissa kaikkialla ja koko ajan. Arjen esineet ja koneet viestivät langattomasti keskenään sekä säätävät toimintaansa itsenäisesti.
Sulautettu järjestelmä (engl. *embedded system*)	Sulautetun järjestelmän määritelmä on Tero ja Kimmo Karvisen mukaan ”järjestelmä joka on rajattuun tarkoitukseen tehty laite, jossa on pieni tietokone. Sulautetut järjestelmät ovat tyypillisesti reaktiivisia järjestelmiä. Ne ovat jatkuvasti interaktiossa ympäristönsä kanssa ja reagoivat ympäristön määräämällä tahdilla. ” (Karvinen & Karvinen 2009, 9.) Sulautettu järjestelmä on esimerkiksi tekemäni prototyyppi (LIITE 4), jossa itsenäinen laite huomaa muutoksen (postilaatikon kannen avaaminen) ja viestittää tiedon eteenpäin.
Käyttäjä	Puhuttaessa käyttäjästä osana innovaatioprosessia, käytän Eric Hippelin (2006, 3) määritelmää: ”Käyttäjät ovat yksilöitä tai yrityksiä, jotka odottavat hyötyvänsä tuotteen tai palvelun käytöstä, vastakohtana valmistajille, jotka odottavat hyötyvänsä tuotteen tai palvelun myymisestä. ”Hippelin (2006, 2, 33–34) mukaan käyttäjät kehittävät innovaatioita omiin tarpeisiinsa. Prosessia ei rajoita tarve ”omistaa ”syntynyttä uutta tietoa esim. patenttien muodossa ja hyötyä siitä rahallisesti.
Käyttäjävetoinen innovaatio (engl. *user-driven/user-dominated/user-centered innovation*)	Innovaatioprosessi, jossa käyttäjä toteaa tarpeen uudistukselle, kehittää mallin parannuksesta, rakentaa siitä mahdollisesti prototyypin tai toimintatavan ja testaa sitä käytännössä. Termiä on käytetty synonyyminä käyttäjäkeskeiselle innovaatiolle. Minä näkisin vivahde-eron termien välillä. Mielestäni *käyttäjäkeskeinen* kuvaa paremmin yritysten tekemää, käyttäjän tarpeita huomioivaa kehitystä. Käytän siis käyttäjävetoista innovaatiota tehdäkseni eron innovoivan loppukäyttäjän ja tuotekehitykseen osallistuvan käyttäjän välille.
Edelläkävijäkäyttäjä (engl. *lead user*)	Edelläkävijäkäyttäjät ovat käyttäjiä, jotka tiedostavat tarpeen jollekin uudistukselle ennen muita ja yrittävät usein vastata tähän tarpeeseen omilla innovaatioillaan (Hippel 1986, 1).

Innovaatioiden diffuusio (engl. *diffusion of innovations*)	Innovaatioiden diffuusiolla tarkoitetaan uudistuksen vastaanottoa ja leviämistä käyttäjien keskuudessa tiettyjen kanavien kautta, tietyn ajan kuluessa (Rogers 2003, 11).
Suunnittelutila (engl. *design space*. suomennos Saarinen 2009, 6)	Suunnittelutila syntyy, kun joku käyttäjä löytää mahdollisuuden tiettyyn kontekstiin sidotulle uudistukselle. Jokainen suunnittelutilaan kehitetty uudistus voi vuorostaan laajentaa sitä tai avata aivan uusia suunnittelutiloja. Yksinkertaisena esimerkkinä oma tapani käyttää juustohöylää. Pidän kurkkusiivuista leivälläni, mutta niiden viipaloiminen ohuiksi ja tasapaksuiksi viipaleiksi veitsellä on työlästä. Ratkaisuksi keksin viipaloida kurkkua juustohöylällä, jolloin pieni suunnittelutila syntyi. Joku tuttuni voi huomata tapani käyttää juustohöylää ja keksiä sille uusia vastaavia käyttötapoja. Myönnettäköön että maailmassa on hyvin rajallinen määrä järkeviä kohteita juustohöylän käyttöön, eli suunnittelutila tulisi hyvin nopeasti tyhjennetyksi.
Edullinen innovaatiotila (engl. *low cost innovation niche*)	Termi kuvaa sitä tilaa, jossa yksittäisillä käyttäjillä on suurimmat mahdollisuudet innovaatioiden tekemiseen. Edullinen tarkoittaa ulkopuolisen tarve- ja ratkaisutiedon ja muiden ulkopuolisten resurssien mahdollisimman vähäistä tarvetta. Tiedon "tahmeus" ja sen siirtämisen kustannukset voivat rajoittaa edullisen innovaatiotilan muodostumista, mikäli ulkopuolista tietoa tarvitaan. Tästä syystä käyttäjät luottavat yleensä omiin resursseihinsa ongelmanratkaisussa (Hippel 2006, 70). Käyttäjät innovoivat elämänaloilla, joissa heillä on aiempaa kokemusta tai tarvetietoa ja jokaisen käyttäjän edullinen innovaatiotila on erilainen. (Hippel 2006, 74.)
Tarvetieto (engl. *need-information*)	Tarvetieto kuvaa tarvetta uudistukselle, johon innovaatiolla yritetään vastata. Jokaisella käyttäjällä voi olla erilaisia tietoja uudistusten tarpeesta ja niinpä tarvetieto on usein hyvin käyttäjäkohtaista.
Ratkaisutieto (engl. *solution-information*)	Siinä missä tarvetieto kertoo uudistuksen tavoitteesta, ratkaisutieto puolestaan vastaa ongelmiin, joita kohdataan matkalla toimivaan innovaatioon. Ratkaisutieto voi olla hyvinkin yksinkertaista, esimerkiksi juustohöylän käyttö uudella tavalla. Kaikkia ongelmia, joita innovaatio- ja prototyyppiprosessissa kohdataan, voidaan pitää ratkaisutietoa vaativina. Tarvittavan ulkopuolisen tiedon määrä ja laatu riippuvat innovaatiosta ja käyttäjän resursseista.
Tahmea tieto (*sticky information*)	Tahmealla tiedolla tarkoitetaan tietoa, joka ei ole siirrettävissä vastaanottajalle helposti hyödynnettävässä muodossa ilman suuria kustannuksia. Tuotekehitys- ja innovaatioprosessissa suuri osa tarvittavasta tietoa voi olla tahmeaa. (Hippel 2006, 66–68.) Esimerkki. Onnistunut tuotekehitys vaatii käyttökontekstin syvällistä tuntemista. Kehittäjät tarvitsevat tarvetietoa, jota on tuotteen tulevilla käyttäjillä. Tieto täytyy siirtää käyttäjiltä kehittäjille, jotta sitä voidaan hyödyntää. Tiedon siirtämisen "hinnalla" ei tarkoiteta vain rahallista kustannusta, vaan myös aikaa ja muita käytettyjä resursseja. Mitä korkeammat tiedon siirtämisen kustannukset ovat, sitä tahmeampana voidaan tietoa pitää.

3 PROTOTYYPIT JA TUTKIMUS

Tein 4 elektronista prototyyppiä (liitteet 1-4), joiden luomisprosessin dokumentoin kehityspäiväkirjan, valokuvien ja tietolähteiden avulla. Dokumentointia ja prosessia analysoiden tutkin, mitkä tekijät olivat prosessin kannalta haasteellisia, mikä oli hyödyllistä ja mitä voitaisiin parantaa, jotta käyttäjän innovaatiokyky kasvaisi entisestään. Dokumentointi kuvastaa myös henkilökohtaista oppimisprosessia.

Tarvetieto prototyyppeihin tuli minulta itseltäni, ja ratkaisutiedon lähteitä olen kirjannut ylös merkittävimpien ongelmien kohdalla. Oma tietotasoni vaikutti tarvittavan ratkaisutiedon määrään ja laatuun. Lähtötasoni oli monessa suhteessa matala, enkä pidä itseäni edelläkävijäkäyttäjänä siinä merkityksessä kuin Hippel (1986, 1) on termin kuvannut. Eräs työni tarkoituksista onkin tutkia, vaaditaanko käyttäjältä omaa syvällistä tietoa innovaatioiden tekemiseksi, eli onko muutos todella voimauttanut käyttäjiä. Prototyyppiprosesseja edeltävää tiedonhakua en ole kirjannut ylös. Olisi lähes mahdotonta erottaa vaikkapa aiemman ohjelmointikokemukseni tai elektroniikan tuntemuksen vaikutusta tarvittavan uuden tiedon määrään ja laatuun.

3.1 Prototyypit

Ensimmäinen tekemäni prototyyppi vastasi tarpeeseeni käyttää biometristä syötettä monipuolisemmin kuin se oli aiemmin mahdollista. Projektin biometrinen syöte oli sykemittarin lähettämä tieto, jonka luin tietokoneelle ja muokkasin käytettäväksi muissakin sovelluksissa. Huomionarvoista projektissa oli toisen käyttäjän luoma suunnittelutila ja kaupallistettu innovaatio, joka vastasi minun tarpeeseeni ja ratkaisi suuren osan ratkaisutiedon ongelmista.

Toinen tekemäni prototyyppi perustui ajatukseen yksinkertaisesta ”kotivahdista”. Kotivahdin idea oli olla etäkäytettävissä oleva hallintapaneeli, jolla voidaan ohjata elektronisia laitteita. Tavoitteena oli saada asunto asutun oloiseksi. Huomionarvoista tässä projektissa oli vaaditun ratkaisutiedon suuri määrä ja tiedon tahmeus, joka lopulta esti toimivan prototyypin valmistumisen. Valmistuessaan prototyypillä olisi ollut mahdollisuuksia avata runsaasti uusia suunnittelutiloja.

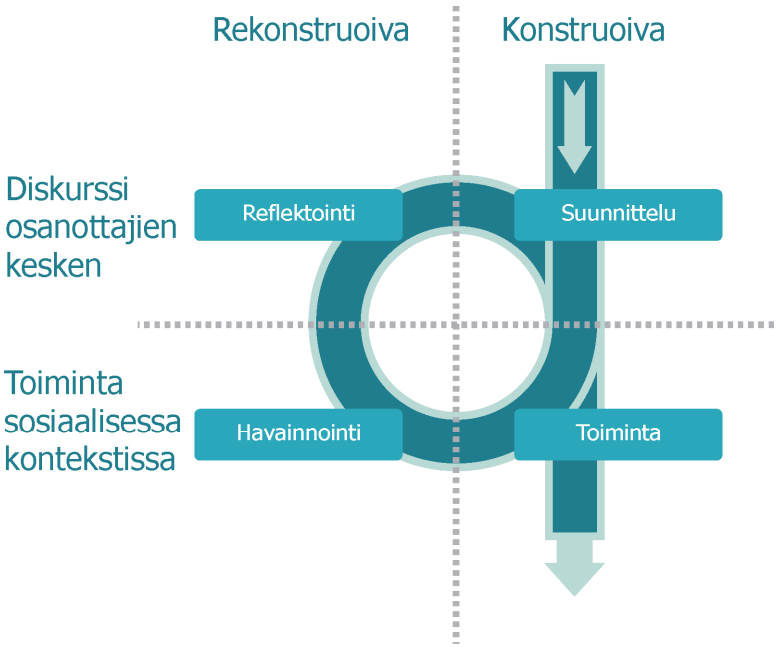
Kolmannen tekemäni prototyypin tavoitteena oli luoda konenäköön perustuva käyttöliittymä, ”köyhän miehen piirtopöytä”. Tarkoituksena oli tehdä sovellus, joka olisi lähes kaikkien rakennettavissa ja jonka tarpeet löytyvät lähes jokaisesta kodista. Projektissa käytin hyödykseni valmiita ohjelmointikirjastoja uudella tavalla ja työn hyötyarvo lienee enemmänkin avautuneissa sovellusmahdollisuuksissa kuin itse ”piirtopöydässä”.

Neljäs prototyyppi perustui ideaan välittää tieto fyysisestä muutoksesta näköpiirin ulkopuolella nähtäväksi. Tässä tapauksessa halusin tiedon siitä, että postilaatikkooni on saapunut postia. Näennäisen yksinkertainen projekti tarjosi yllättävän paljon ratkaisutiedon haasteita, joita en kyennyt ennalta arvioimaan.

3.2 Toiminnallinen kehittäminen

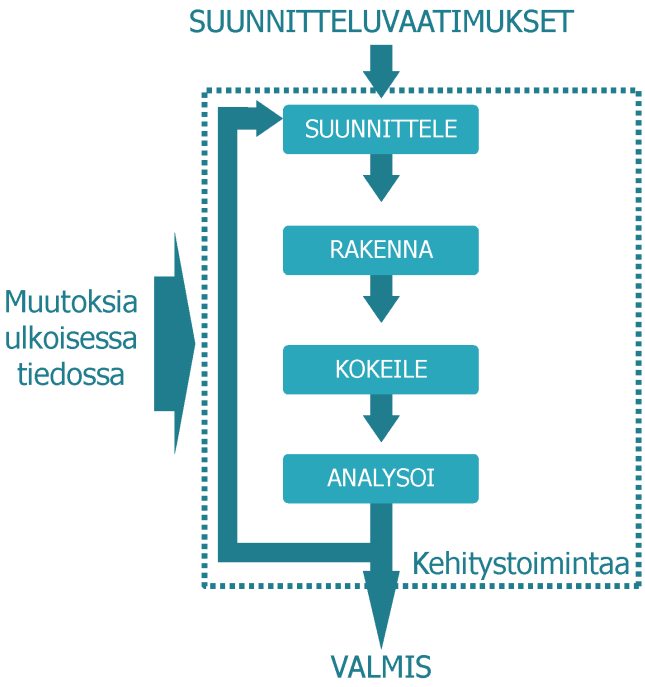
Valitsin tutkimukseeni osallistuvan lähestymistavan saadakseni kokonaisvaltaisen käsityksen käyttäjän roolista innovaatioprosessissa. Osallistuva havainnointi tai toimintatutkimus sopi laadullisen ja kokemusperäisen tiedon keräämiseen. Toimintatutkimuksesta on monenlaisia tulkintoja ja näkemyseroja. Useat kuvaukset toimintatutkimuksesta kuitenkin painottavat sen yhteisöllistä aspektia (Eskola & Suoranta 1998, 98; Heikkinen 2007, 205–208). Minä en tutkinut yhteisöjen toimintaa osallistumalla suoraan niiden toimintaan, vaan tarkastelemalla niiden merkitystä mahdollisuuksiini kehittää innovaatioita.

Tein tutkimusta omasta työstäni, jolloin tarkkailijan ja kohteen roolit yhdistyivät. Tämä dualistinen asema tutkimuksessa on mahdollista itsereflektion avulla. Heikkisen mukaan reflektion keskeisyys toimintatutkimuksessa ilmenee tutkimuksen hahmottamisena itsereflektiivisenä kehänä (kuva 1).



Kuva 1. Toimintatutkimuksen vaiheet. Alkuperäinen kuva (Carr & Kemmis 1986, 186).

Kehässä toiminta, sen havainnointi, reflektointi ja uudelleensuunnittelu seuraavat toisiaan (Heikkinen 2007, 202.) Tämä kehämalli sopi hyvin osaksi kehitys- ja tutkimusprosessia, sillä se vastaa lähes täysin yritys-ja-erehdys kehämallia[1] (kuva 2).

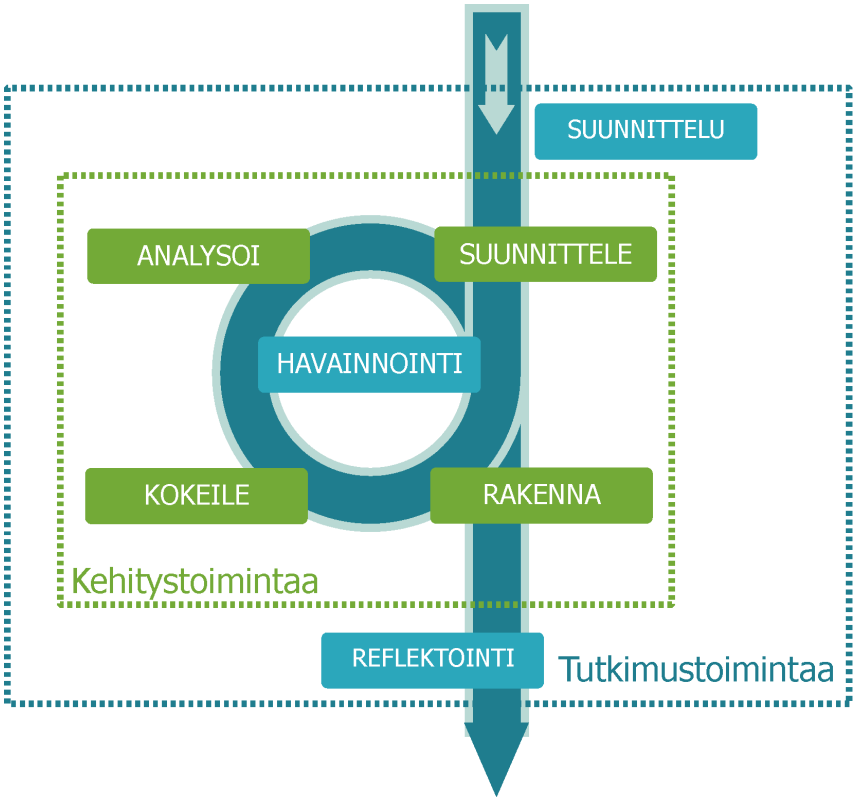


Kuva 2. Yritys-ja-erehdys kehän kuvaus. Alkuperäinen kuva (Hippel 2006, 64).

Ensimmäisen prototyypin aikana toteutin jatkuvaa yhtäaikaista prototyyppikehitystä, havainnointia ja prototyypin analysointia, sekä tutkimuksen reflektointia. Voisi sanoa, että yritin mahduttaa kehittämisen ja tutkimuksen samanaikaisesti prosessiin. Arvioin kokemuksieni perusteella mallin uudestaan ja jaoin tutkimukseni ja prototyypin kehityksen erillisiksi tehtäviksi. Uudessa mallissa havainnointi on ainoa tutkimuksellinen tehtävä, jota suoritan samanaikaisesti prototyypin kehityksen aikana. Käytännössä keskeytin kehitysprosessin tehdäkseni muistiinpanoja kehityksen kulusta, tiedon lähteistä ja muista havainnoista. Samalla dokumentoin prototyypin edistymistä kehityspäiväkirjan avulla. Yhdistämällä prototyypin kehitysprosessin ja toimintatutkimuksen mallit, loin toiminnallista tutkimustani kuvaavan kaavion (kuva 3). Toimintatavan heikkous on ihmisen keskittymiskyvyn rajallisuudessa. Monet prototyyppeihin liittyvät ongelmat vaativat jakamattoman huomioni pitkäksi aikaa

[1] engl. Trial-and-error.

kerrallaan, jolloin tutkimuksellinen havainnointi ja dokumentointi jäivät vähemmälle. Tämä oli kuitenkin kokemukseni mukaan paras tapa saavuttaa toimivia prototyyppejä.



Kuva 3. Muokattu toiminnallisen kehityksen kuvaus.

Koin tutkimusmallin kaksoisroolin hetkittäin haasteelliseksi. Yritys olla vaikuttamatta tutkijan tuloksiin ollessani kehittäjän roolissa, oli jo itsessään tuloksiin vaikuttamista. Kuten kaikkea kokemusperäistä tietoa, saavutettua tietoa käsitellään henkilökohtaisen tulkinnan ja arvomaailman kautta. Näin ollen tekijä-tutkijan saavuttama tieto ei voi olla objektiivista sanan tavanomaisessa merkityksessä (Heikkinen 2007, 196–205.) Koin kuitenkin toiminnallisen kehittämisen ainoaksi malliksi, jolla saisin kokonaisvaltaisen kuvan käyttäjän asemasta.

4 INNOVAATIO

Kehittääkseni innovaatioita minun piti ensin selvittää mitä sanalla todella tarkoitetaan. Everestt Rogersin (2003, 11–12) mukaan innovaation eli uudennoksen määritelmä on jokin uusi tuotos. Se voi olla idea, käytäntö tai esine, jota pidetään

ominaisuuksiltaan uutena. Tekesin (2006) määritelmän mukaan "Innovaatio tarkoittaa kaupallisesti tai yhteiskunnallisesti uudella tavalla hyödynnettyä tietoa ja osaamista."

Innovaation ei tarvitse perustua täysin uuteen tekniseen ominaisuuteen tai tietoon ollakseen innovaatio, vaan uuteen tapaan hyödyntää niitä. Rogers (2003, 12) kuvaa uutuuden suhteellisuutta seuraavasti. Idean objektiivinen uutuus ajallisesti mitattuna sen keksimisestä tai ensimmäisestä käyttökerrasta ei ole oleellista. Tärkeää on idean käsitteellinen uutuus käyttäjälle, mikä määrittää hänen reaktionsa siihen. Mikäli idea vaikuttaa uudelta, se on innovaatio. Monia pieniä tai suurempia parannuksia jo olemassa oleviin tuotteisiin voidaan myös pitää innovaatioina. Mikä on rullalauta? Se on potkulauta, jonka ohjaustanko on katkennut. Mikä on lumilauta? Se on monta laskettelusuksea yhdistettynä vierekkäin. Monet innovaatiot saavat alkunsa käyttäjien "virheellisestä (muu kuin alkuperäinen käyttötarkoitus)" tai uudenlaisesta tavasta käyttää tuotteita ja alkavat elää omaa elämäänsä (DiBona & Cooper & Stone 2005, 341).

4.1 Esimerkki innovaatioiden syntymisestä

Selventääkseni työssä käytettyjä termejä ja niiden välisiä suhteita kuvaan innovaatioiden syntymistä tarinan avulla.

Ensimmäiset pioneerit (edelläkävijäkäyttäjät) ovat löytäneet alueelta kultaa ja ennen pitkää paikalle saapuu muitakin kullankaivajia (muut käyttäjät). Pioneerit löysivät paikan ensimmäisinä, joten heillä oli enemmän aikaa tutustua maastoon ja kullankaivamisen haasteisiin. Eräs pioneereista oli pesemässä ruokalautastaan puron varrella ja havaitsi, kuinka lautasen pohjalle kertyi sattumalta hiekkaa painavampia kultahippuja. Tämä vahinko loi uudenlaisen tavan etsiä kultaa (eli innovaation, käyttäjän "väärä" tapa käyttää lautasta johti uuden suunnittelutilan löytämiseen). Ennen pitkää tieto uudesta tavasta levisi muidenkin kullankaivajien keskuuteen (innovaation diffuusio). Keraamisilla lautasilla oli kuitenkin paha tapa rikkoontua kivikkoisissa puronuomissa. Kaikki lautasensa rikkonut kullankaivaja tiesi tarvitsevansa kestävämpiä lautasia (tarvetieto). Hän muotoili saatavilla olevista materiaaleista (edullinen innovaatiotila) eli ohuesta metallilevystä (ratkaisutieto) kestävämmän "lautasen" (innovaation diffuusio ja suunnittelutila mahdollistivat uuden innovaation).

Tehokkaamman kullanetsintätavan levitessä kaivajien keskuuteen ja uusien kaivajien saapuessa paikalle, kultaa jäi yhä vähemmän uusille tulijoille (suunnittelutilan tyhjentyminen, engl. *mining out the design space*. Hippel 2006, 21–22). Jotkut pioneereista totesivat kaivamisen heikentyneen hyödyn ja alkoivat sen sijaan myydä kehittelemiään "metallilautasia" uusille tulijoille (muutos käyttäjä-innovaattorista käyttäjä-valmistajaksi).

Tämä yksinkertaistettu esimerkki kuvaa käyttäjäinnovaatioiden luonnetta. Ne syntyvät kokemusperäisestä tarpeesta ja niihin vastataan usein juuri silloin käytössä olevilla resursseilla. Lyhyesti sanottuna, suurin osa käyttäjäinnovaatioista on omaan tarpeeseen keksittyjä "patenttiratkaisuja," joilla saattaa olla käyttöarvoa muille vastaavassa tilanteessa oleville käyttäjille.

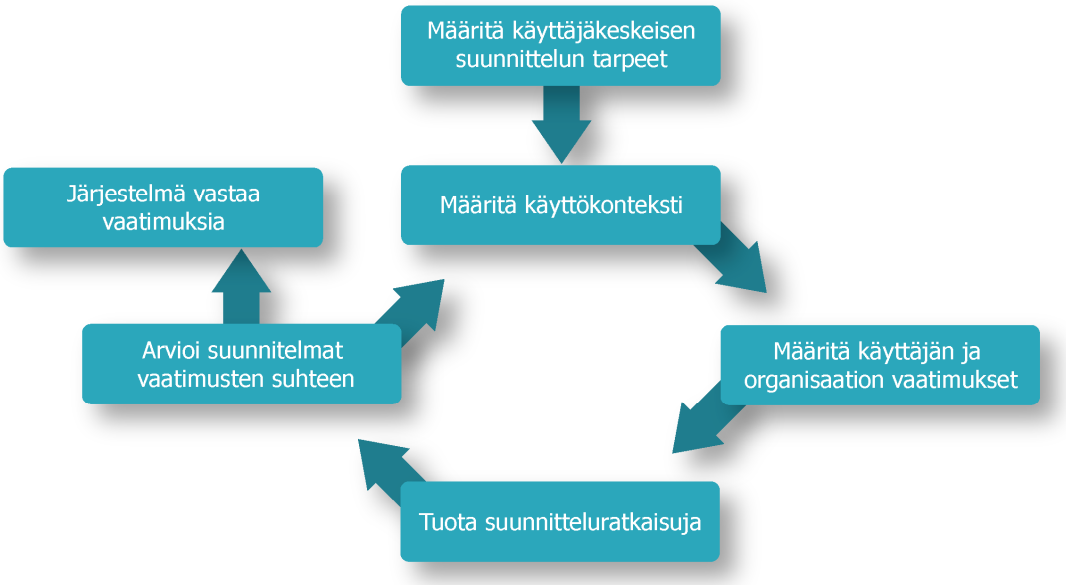
4.2 Mistä innovaatio muodostuu?

Yksinkertaisimmillaan onnistuneeseen innovaatioon tarvitaan oikea tieto, oikean henkilön käytettäväksi oikeaan aikaan. Innovaatioiden lähtökohtana on usein ongelma tai tarve, johon yritetään etsiä ratkaisua. Tarvetieto kuvaa nimensä mukaisesti tätä tarvetta, johon innovaatiolla yritetään vastata. Usein tarpeeseen vastaavan ratkaisun kehittäminen vaatii tietoa tai taitoa. Tätä tietoa kutsutaan ratkaisutiedoksi. Innovaatiot kumpuavat näistä tiedoista, mutta niiden konkretisoiminen vaatii usein resursseja, eli esimerkiksi työkaluja, pääomaa ja aikaa. Käytän innovaatioprosessin jakamiseksi tarve- ja ratkaisutietoihin Eric Von Hippelin kuvausta (2006, 8, 147–149). Tarve- ja ratkaisutiedot ovat näennäisen erillisiä käsitteitä, mutta niiden erottaminen on käytännössä hankalaa. Kontekstikohtainen ongelma voi vaatia omanlaisensa uniikin ratkaisun, jolloin tiedot ovat läheisesti sidoksissa toisiinsa.

4.2.1 Tarvetieto

Innovaatio vastaa havaittuun tarpeeseen tai tuottaa sellaisen uusien ominaisuuksiensa myötä. Tutkimuksen kautta tuotetun tekniikan tai keksinnön arvo voi olla melko olematon, jos sille ei löydetä käytännön sovelluksia, niinpä suuri osa innovaatioista lähtee liikkeelle tarpeiden kartoituksesta. Tarvetieto kuvaa tarvittavaa tietoa siitä

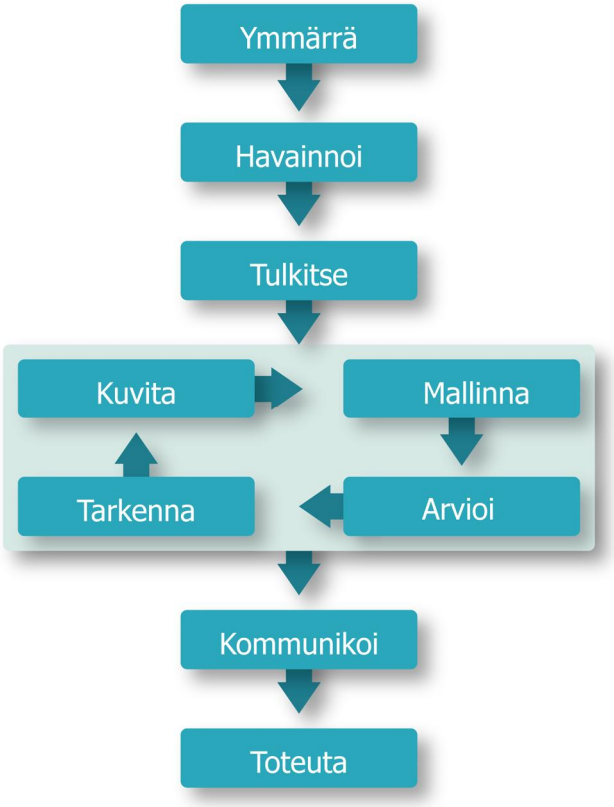
tarpeesta johon innovaatio vastaa. Innovaatio- ja tuotekehitysprosesseille on olemassa laaja kirjo prosessikuvauksia ja ohjeistuksia, esimerkkinä tarvetiedon merkityksestä ISO 13407 standardin mukainen prosessikuvaus, jossa ensimmäinen askel on tavoitteiden ja tarpeiden kartoittaminen (kuva 4).



Kuva 4. ISO 13407 standardi. Vuorovaikutteisten järjestelmien käyttäjäkeskeinen suunnitteluprosessi. Alkuperäinen lähde ISO-organisaatio.

Toinen esimerkki tarvetiedon merkityksestä on menestyksekkään IDEO-yrityksen tuotekonseptoinnin ja innovoinnin prosessikuvaus (kuva 5). IDEO on innovaatio- ja suunnittelupalveluita tarjoava konsulttiyhtiö, joka on ollut osallisena mm. Applen ensimmäisen hiiren suunnittelussa ja päässyt BusinessWeek- lehden 25 parhaan innovaatioyrityksen listalle. Yhtiön menestyksestä kertoo myös se, että yhtiö tekee konsultointia listan muille 24 yritykselle.

Molemmat prosessikuvaukset kertovat tarvetiedon ja kontekstin ymmärtämisen merkityksestä innovaatioiden kehityttämisessä. Yritykset yrittävät mallintaa käyttäjän toimintaa ja ymmärtää kehitettävän tuotteen käyttökontekstia. Käyttäjällä, joka kehittää omia ratkaisuitaan on se etulyöntiasema, että hän tietää mitä haluaa ja milloin kehitetty tuote tai palvelu vastaa vaatimuksia.



Kuva 5. IDEOn prosessikuvaus. (Alkuperäinen malli Keinonen & Jääskö 2003, 57).

4.2.2 Ratkaisutieto

Tieto siitä mitä tarvitaan voi olla hyvä lähtökohta innovaatiolle, mutta useimmissa tapauksissa tarvitaan vielä erikoistunutta osaamista innovaation konkretisoimiseksi. Ratkaisutieto voi olla teknistä osaamista, muunlaista henkistä pääomaa, kädentaito tai käyttökontekstista riippuva toteutustapa tai tieto, joka voi olla hyvin lähellä tarvetietoa. Ratkaisutieto voi esimerkiksi olla koodinpätkä tai tieto siitä, että kyseinen koodinpätkä toimii tai ei toimi halutulla tavalla tietyssä yhteydessä. Ratkaisutiedon lähteitä on varmasti yhtä monta kuin ongelmiakin, kontekstista ja ongelmasta riippuen.

4.3 Tiedon epätasapaino ja tahmeus

Jokainen innovaatioprosessi tarvitsee sopivat tiedot onnistuakseen. Jos tarvittavaa tietoa ei ole, sitä pitää tuoda ulkoapäin. Tarve- tai ratkaisutiedon puuttuminen voi johtaa tiedon epätasapainoon, jolloin prosessin lopputulos voi jäädä heikoksi (Hippel

2006, 8, 66—67). Vaillinaiseen tietoon perustuvaan kehitykseen pätee totuus ”roskaa sisään, roskaa ulos”. Tietojen erilaisuus ja niiden sijainti luovat prosessille haasteita. Voidaan yleistää, että käyttäjillä on hallussaan syvällisempää tarvetietoa kuin valmistajilla. Valmistajilla on puolestaan syvällisempää ratkaisutietoa. Tarvittavan tiedon siirtämisen kustannukset voivat tehdä tiedosta tahmeaa ja vaikeuttaa prosessia.

Tarvetiedon ollessa pääasiallisesti lähtöisin käyttäjältä itseltään ratkaisutieto nousee useimmiten suuremmaksi kynnykseksi käyttäjäinnovaation onnistumiselle. Tekemissäni prototyypeissä ratkaisutiedon hankinta oli ratkaisevassa asemassa. Hankinnan kustannukset käytettyinä resursseina kuvastivat melko hyvin tarvittavan tiedon määrää ja tahmeutta kussakin projektissa.

5 EDULLISEN INNOVAATIOTILAN LUOMINEN

Käyttäjien kyky innovoida on suurimmillaan niillä aloilla, joista heillä on aikaisempaa kokemuspohjaista tietoa, työn tai harrastuksien kautta (Hippel 2006, 74). Tällaista erikoisosaamisen aluetta Hippel kutsuu edulliseksi innovaatiotilaksi. Näillä alueilla käyttäjien mahdollisuudet innovoida ovat vahvimmillaan, koska heillä on kokemuksen kautta tarvetietoa jollekin uudistukselle ja he saattavat keksiä siihen toimivan ratkaisun. Usein käyttäjäinnovaatiot syntyvät juuri edullisen innovaatiotilan kautta, koska käyttäjät tekevät uudistuksia omiin tarpeisiinsa. Tarve taas tulee kokemuksesta jota käyttäjät ovat saaneet työn tai harrastuksen kautta.

Suunnittelutila on käsite, jolla tarkoitetaan käyttäjien havaitsemaa mahdollisuutta innovaatioihin (Baldwin, Hienerth & Hippel 2006, 21). Tämä voi tarkoittaa vaikkapa jonkin olemassa olevan tuotteen käyttämistä uudella tavalla tai parannuksia olemassa oleviin tuotteisiin. Esimerkki suunnittelutilan syntymisestä on tapaus jossa käyttäjät hakkeroivat Applen iPhonen. Hakkerointi mahdollisti kolmannen osapuolen ohjelmien ajamisen laitteella, jolloin ”uusi” alusta tarjosi käyttäjille uuden suunnittelutilan. Myöhemmin Apple taipui käyttäjien vaatimuksiin ja avasi *approved apps* palvelun, jossa sovellusten levittäminen oli mahdollista. (Wilson, 2010, 39.)

Aloitin prototyyppien rakentamisen kartoittamalla tarvitsemani resurssit. Voidakseni menestyksekkäästi kehittää innovaatioitani, minun piti ”rakentaa” itselleni edullinen

innovaatiotila. Tämä tarkoitti omien resurssieni tiedostamista ja hyödyntämistä. Vaikka minulla on kokemusta tietotekniikasta ja ohjelmoinnista, en tiennyt juuri mitään elektroniikasta tai sähköopista. Nämä olivat ne alueet joissa tahmean tiedon siirto vaatisi resursseja. Kaavailemani prototyypit sisälsivät yhtä lukuun ottamatta kaikki elektronisia komponentteja. Puutteellisten tietojeni vuoksi en pidä itseäni edelläkävijäkäyttäjänä. Tämä sopi hyvin koska tarkoituksenani oli tutkia, onko "tavallisella" käyttäjällä mahdollisuuksia innovaatioiden tekemiseen, eli onko ympäristömme muutos voimauttanut käyttäjiä.

5.1 Tiedon lähteet ja tahmean tiedon siirtäminen

Innovaatiotilani ei ollut aluksi edullinen, joten jouduin turvautumaan suuresti ulkopuoliseen tietoon. Aloitin etsimällä tiedon lähteitä ja apukeinoja tiedon tahmeuden poistamiseksi. Suurin osa tarvitsemistani ratkaisutiedoista koski kehitystyökalujani ja elektronisia komponentteja. Aiheesta löytyi verkosta valtavasti tietoa. Tiedon hyödynnettävyys oli monissa tapauksissa kuitenkin kyseenalaista. Tiedon siirtämisen kustannukset muodostuivat sen etsimiseen kuluneesta ajasta, tiedon muokkaamisesta projektiin sopivaan muotoon ja mahdollisista taloudellisista kuluista. Taloudellisia kuluja tiedon hankkimiseen kului yllättävän vähän ja nekin olisi voitu välttää valitsemalla avoimia tiedonlähteitä kirjojen sijaan.

Tiedon etsimiseen kului runsaasti aikaa. Tähän oli pääsyynä tiedon sirpaloituminen ja toisaalta sen runsaus. Löytääkseni sopivan tiedon helposti hyödynnettävässä muodossa jouduin käymään läpi lukuisia tiedonlähteitä. Parhaimmassakin tapauksessa tieto oli silti osittain tahmeaa, sillä minun piti muokata löydetty tieto omiin projekteihini sopivaksi. "Melkein sopivia" ratkaisutietoja oli kuitenkin runsaasti saatavilla. Millä keinoin käyttäjä voi laskea tiedon siirron kustannuksia? Yksi tapa on käyttää tarkoitukseen suunniteltuja kehitystyökaluja. Ne laskevat tarvittavan ratkaisutiedon tasoa ja määrää. Toinen tapa on etsiä juuri sopiva tieto, jolloin sen etsimiseen kulutettu aika voi tosin muodostua suureksi kustannukseksi. Kolmas tapa on turvautua johonkin ulkoiseen tahoon, joka on jo ratkaissut kyseisen ongelman tai voi ratkaista sen käyttäjän puolesta. Projekteissani pyrin ratkaisemaan ongelmat aina yksinkertaisimmalla ja nopeimmalla mahdollisimmalla tavalla.

5.2 Työkaluja käyttäjäinnovaatioon ja räätälöityyn tuotesuunnitteluun.

Käyttäjien tarpeisiin on kehitetty innovaation ja tuotesuunnittelun työkaluja, jotka mahdollistavat käyttäjän ajatusten konkretisoitumisen prototyypeiksi tai toimiviksi tuotteiksi. Jakamalla innovaatioprosessi tarve- ja ratkaisutietopohjaisiin alitehtäviin voidaan nämä tehtävät ratkaista erillään. Käyttäjille suunnattujen innovaatio- ja suunnittelutyökalujen ajatus pohjautuu tietojen eriyttämiseen ja ratkaisutiedon tarjoamiseen, teknisten apuvälineiden kautta. (Hippel 2006, 16, 147.)

Tietojen eriyttämisen taustalla on ymmärrys siitä, että tiedon siirtäminen voi tehdä siitä tahmeaa ja tietoa kannattaa soveltaa siellä missä se on luotu. Esimerkiksi tuotetta suunnittelevan käyttäjän voi olla vaikea välittää yksityiskohtaista tietoa haluamistaan ominaisuuksista jollekin toiselle taholle. Tämä tieto voi olla kokemusperäistä ja vaikeasti mitattavaa. Käyttäjän ja mahdollisen valmistajan kannalta on taloudellisempaa tarjota käyttäjälle helppokäyttöiset suunnittelutyökalut, joilla käyttäjä voi kokeilla kunnes on tyytyväinen lopputulokseen. Tämän jälkeen työkalun avulla siirretään käyttäjän tarvetieto valmistajalle, jolla on kyky ratkaisutiedon tarjoamiseen.

Etsi projektejani varten työkaluja, jotka vapauttaisivat minut elektronisten laitteiden käytön ratkaisutiedoista ja mahdollistaisivat monipuolisen prototyypittämisen. Havaitsin, että suunnittelutyökaluja löytyy niin suljetun kuin avoimen lähdekoodin tekijöiltä. Avoimen lähdekoodin yhteisöt ovat kehittäneet työkaluja omiin tarpeisiinsa. Tämä sopi minulle hyvin, sillä ilmaiset kehitysympäristöt laskisivat kehityskustannuksia. Tein vertailua yleisimpien elektronisten kehitysalustojen välillä, käyttäen hyväkseni Hippelin (2006, 154) listaamaa viittä ominaisuutta, jotka suunnittelutyökalun tulisi sisältää, jotta sitä voidaan pitää korkealaatuisena.

1. työkalu mahdollistaa käyttäjälle yritys & erehdys -tyyppisen kokeilevan kehitysmallin. ”Virheitä saa tehdä.”
2. työkalu tarjoaa käyttäjälle tarvittavan vapauden ja ohjaa suunnitteluprosessissa haluttuun suuntaan.
3. työkalun tulee olla helposti omaksuttava ilman erikoista koulutusta.
4. modulaarinen laajennettavuus.
5. valmiin suunnitelman tulee olla tuotantokelpoinen ja valmistajan tuotantolaitteiston kanssa yhteensopiva ilman muokkauksia.

Elektroniseksi prototyyppilevyksi/kehitysalustaksi valitsin avoimen lähdekoodin Arduinon. Arduino on prototyyppilevy ja ohjelmointiympäristö, joka perustuu Processing-ympäristöön ja on sen kanssa yhteensopiva. Minulla oli perustietoja Processing-ympäristöstä, joten Arduino soveltui hyvin edullisen innovaatiotilani työkaluksi. Arduino tuntui olevan myös suosituin käytössä olevista kehitysalustoista ja sillä on aktiivinen käyttäjäyhteisö, joten ajattelin sen helpottavan ratkaisutietojen löytämistä. Tärkeä valintakriteeri oli myös monipuolinen laajennettavuus, joka Arduinossa ilmenee toiminnallisuutta lisäävien laajennusosien, ns. kilpien[2] ja kirjastojen muodossa.

Muiden Hippelin määrittämien ominaisuuksien paikkansapitävyyttä Arduinon kanssa en tiennyt ennen käytännön kokeilua. Arduino osoittautui helposti opittavaksi ja selkeäksi työkaluksi nopeaan prototyypittämiseen. Alusta mahdollisti kokeellisen kehityksen, jossa virheistä saatu palaute vei prosessia eteenpäin. Käyttämällä valmista elektronista kehitysalustaa minulta itseltäni vaadittavan ratkaisutiedon taso laski huomattavasti.

Modulaarinen laajennettavuus on Arduinon vahvimpia puolia, mikä mahdollisti radiolähettimien käytön kahdessa prototyypissä (LIITTEET 2, 4). En usko, että olisin onnistunut rakentamaan vastaavaa järjestelmää ilman valmista laajennusosaa. Arduinon kirjastot lisäävät toiminnallisuutta valtavasti. Ikävä kyllä kehitysympäristöstä ei löydy apua erillisten kirjastojen käyttämiseen samassa ohjelmassa. Toimintalogiikan rakentaminen ja kirjastojen toiminnallisuuden yhdistäminen jää käyttäjän harteille. Tällaisten ratkaisutietojen hankkiminen oli vaikeaa ja vaati paljon yritys-ja-erehdys-tyyppistä kokeilua. Näissä tapauksissa jouduin turvautumaan omaan tietooni ohjelmoinnista.

Kehitysalustan suosio vaikuttaa sen hyödynnettävyyteen. Mitä useammalla käyttäjällä on samat kehitystyökalut käytettävissään, sitä todennäköisempää on, että haettu ratkaisutieto löytyy helposti hyödynnettävässä muodossa. Arduino on suosituin elektroninen kehitysalusta, Helmikuussa 2010 niitä oli myyty jo 130.000 (Tom Igoe, Adafruit-blogi 6.2.2010).

---

[2] engl. Shield. Elektronisia lisäosia, jotka lisäävät kehitysympäristön toiminnallisuutta. Useat toimivat suoraan ilman suurempia muokkauksia.

Ainoat laadukkaan työkalun ominaisuudet, joissa on kokemukseni mukaan parantamista Arduinossa, ovat valmiin suunnitelman tuotantokelpoisuus ja käyttäjän ohjaus oikeaan suuntaan. Arduinon ohjelmointiympäristö ei sisällä mahdollisuutta valmiin tuotteen piirilevyn suunnittelemiseen. Tähän tarkoitukseen on kuitenkin kehitetty Fritzing.

Arduino tarjoaa käyttäjälleen valtavat mahdollisuudet, vaikka kehitysalusta yksinään on hyvin rajallinen laite. Arduinon rikkaus tulee siihen yhdistettävien sensoreiden, komponenttien ja ohjelmoinnin loputtomista yhdistelmistä. Tämä monipuolisuus lisää uusien suunnittelutilojen syntymisen mahdollisuutta ja pienentää todennäköisyyttä, että suunnittelutila tyhjenisi. Kehitysympäristö tukee käyttäjää yksittäisten sensorien tai lähtöjen[3] hallinnassa valmiiden esimerkkien avulla, mutta eri osien loogiset yhteydet jäävät käyttäjän selvitettäviksi. Arduino-piirilevy myös ohjaa käyttäjää oikeaan suuntaan fyysisten elementtien suhteen, sillä selkeästi merkityt jaottelut levyssä rajoittavat käyttäjän virheellisiä toimintoja.

Fritzing on Arduinon ja Processingin hengessä kehitetty avoimen lähdekoodin elektronisen suunnittelun automaatio-ohjelma. Ohjelman avulla voidaan dokumentoida Arduinon prototyyppien elektroniset piirustukset prototyyppivaiheessa, liittää mukaan processing-koodit sekä suunnitella prototyypistä toteutettava piirilevy. Fritzing tuottaa CAD[4]-ohjelmista tuttuja Gerber-tiedostoja, jotka ovat yhteensopivia piirilevyjen valmistuslaitteistojen kanssa. Ohjelma on tärkeä uudistus yhdenmukaisen projektidokumentoinnin saavuttamiseksi ja suunnitelmien tuotteistamiseksi. Käytännön kokeilu osoitti ohjelman olevan vielä kehitysasteella[5] ja sisältävän runsaasti vikoja. Esimerkiksi automatisoitu piirilevyn suunnittelu vaati niin paljon manuaalista korjaamista, että sen koko hyöty on kyseenalainen käyttäjälle, joka ei ole perehtynyt elektroniikkaan. Ohjelmalla voi kuitenkin toteuttaa selkeitä havaintokuvia projektista muistiinpanojen kanssa, mikä on parempi vaihtoehto kuin vaihtelevan tasoiset valokuvat. Fritzing on askel oikeaan suuntaan dokumentoinnin yhdenmukaistamiseksi.

---

[3] toteuttavat komponentit. Esim. LED-valot, äänisummerit ym. tietoa lähettävät komponentit.

[4] engl. computer aided design

[5] versio 0.3.16b, tarkastettu 5.4.2010. http://fritzing.org/

Processing on avoimen lähdekoodin ohjelmointikieli ja kehitysympäristö. Monipuolinen ohjelmointikieli on tullut suosituksi eritoten fyysistä maailmaa ja tietokoneen toiminnallisuutta yhdistävissä teoksissa. Processingin suurimpia rikkauksia ovat sen kirjastot, joiden kehittämisessä avoimen lähdekoodin yhteisöllä on suuri merkitys.
Koin Processingin sopivaksi prototyypittämiseen erityisesti sen matalan oppimiskynnyksen ja modulaarisen laajennettavuuden vuoksi. Parannettavaa ohjelmassa olisi ainakin syntaksin tarkistuksen ja vianetsintäominaisuuksien parantamisessa. Ohjelmointikieli vaatii jonkin verran perehtymistä ja omaa ratkaisutietoa.

5.3 Tiedon avoin jakaminen ja diffuusio

Käyttäjien halukkuus jakaa omaa tietoaan ja löytämiään suunnittelutiloja on merkittävä tekijä käyttäjäinnovaatioiden mahdollistamisessa. Tämä halukkuus on todistettu monissa aiemmissa tutkimuksissa (Hippel 2006, 77—79; Baldwin, Hienerth & Hippel, 2006, 23, 25; DiBona, Cooper & Stone 2005, 345). Eritoten edelläkävijäkäyttäjien tekemä uraa uurtava työ uusien suunnittelutilojen löytämiseksi on hyödyksi muille käyttäjille (Franke & Hippel 2003, 26). Tekemäni prototyypit olivat mahdollisia ilmaisen ja vapaasti jaetun tiedon vuoksi. Tarvitsemistani ratkaisutiedoista yli puolet oli avoimista lähteistä ja liki kolmasosa suoraan muilta käyttäjiltä (taulukko 1). Tähän vaikutti toki valitsemani kehitystyökalut ja niiden yleisyys. Erityisesti Arduinon ympärille on rakentunut niin sanottu avoin tietoyhteisö. Avoimella tietoyhteisöllä tarkoitan Hippelin (2006, 165–166) määritelmän mukaista yhteisöä, joka koostuu yksilöistä tai organisaatioista. Tämän yhteisön jäsenet kokoontuvat yhteisen, kaikille avoimen tiedon ympärille. Tällainen yhteisö toimii tiedon välittäjänä tallentamalla sitä, kuten esimerkiksi Wikipedia, sekä mahdollisesti yhdistämällä tiedon tarvitsijat ja tiedon haltijat.

Everett M. Rogers kehitti mallin, jossa käyttäjät on jaettu viiteen ryhmään sen mukaan, miten he omaksuvat uudet innovaatiot ja ideat (Rogers 2003, 11).

- Innovoijat (innovators) – 2–3 % väestöstä: uskaliaita, koulutettuja, omaavat useita informaation lähteitä;
- Varhaiset omaksujat (early adopters) – 10–15 % väestöstä: sosiaalisia

johtajia, suosittuja, koulutettuja;

- Varhainen enemmistö (early majority) – 30–35 % väestöstä: vastaanottavaisia tultuaan vakuuttuneiksi innovaation omaksumisen hyödyistä, useita eri sosiaalisia kontakteja;
- Myöhäinen enemmistö (late majority) – 30–35 % väestöstä: skeptisiä, perinteisiä, alempi sosio-ekonominen asema;
- Hitaat omaksujat (laggards) – 10–20 % väestöstä: vastustavat aktiivisesti uusia innovaatioita, naapurit ja ystävät pääasiallinen tiedonlähde, pelkäävät velkaantumista.

Rogersin malli vastaa Hippelin kehittämää teoriaa edelläkävijäkäyttäjistä (Hippel 2006, 22). Nämä käyttäjät ovat oman alansa huippuja ja siten tiedostavat tarpeen uusille innovaatioille ennen muita käyttäjiä. Edelläkävijäkäyttäjät ovat Rogersin jaottelun mukaisesti *innovoijia*. Uskon että omien tuotteiden räätälöinti on kasvava trendi, jolloin Arduinon kaltaisessa erikoistuneessa tietoyhteisössä melkein kaikkia käyttäjiä voidaan pitää vähintään *varhaisena enemmistönä*. Vaikka edelläkävijäkäyttäjät ovat tärkeitä suunnittelutilojen luojia avoimissa tietoyhteisöissä, he eivät ole kuitenkaan ainoita käyttäjiä, jotka kehittävät uusia suunnittelutiloja. Avoimet tietoyhteisöt toimivat tehokkaasti, koska niissä tieto liikkuu vapaasti, jolloin kaikilla on mahdollisuus hyötyä tästä tiedosta ja kehittää sitä. Pienetkin parannukset aiempiin innovaatioihin voivat luoda aivan uudenlaisia suunnittelutiloja (posliinilautasesta metallilautaseen), ja tällaisiin *pienten parannusten innovaatioihin* on melkein kaikilla mahdollisuus. Kaikista innovaatioista suurin osa on juuri pieniä parannuksia aiempiin innovaatioihin (Hippel 2006, 112; 1988, 27, 32).

Huomasin selvästi mitä vaikutuksia tiedon diffuusiolla ja jatkokehityksellä oli avoimissa tietoyhteisöissä. Havaintoni olivat yhteneviä Gwendolyn K. Leen ja Robert E. Colen (2003, 2, 29–30) tekemään tutkimukseen avoimen lähdekoodin yhteisön työstä Linux Kernel- projektin parissa. He huomasivat, että avoimen lähdekoodin yhteisöissä innovaatiota tukevia tekijöitä ovat tekijänoikeudelliset lisenssit, jotka tukevat luottamusta ryhmän sisällä, varmistavat tiedon jakamisen ja paljastavat virheitä tuotoksissa. Avoimen lähdekoodin yhteisö luottaa evolutionaariseen prosessiin, joka perustuu kritiikkiin ja kriittiseen arviointiin. Samat tekijät vaikuttavat myös muun muassa Arduinon yhteisöön. Vaikka Arduinon ympärille kerääntynyt käyttäjäyhteisö ei ole yhtä tiiviisti nivoutunut kun esimerkin projektiryhmä, kaikkia yhdistää silti käytetyn

alustan lähdekoodin avoimuus ja evolutionaarinen kehitysmalli. Huomasin kuinka vanhempia innovaatioita oli jatkokehitelty niiden leviämisen jälkeen. Useat kokeilemani ratkaisutiedot olivat käyttökelpoisia vain, koska muut käyttäjät olivat huomanneet niissä virheitä ja jakaneet tietonsa.

5.4 Havaintoja tiedonlähteistä

Aina kun oli mahdollista, käytin ensisijaisena lähteenä tuotteiden ja palveluiden virallisia dokumentointeja ja esimerkkejä. Yleisesti ottaen kaikki yksittäisten tuotteiden dokumentoinnit olivat riittäviä. Kaupallisten tuotteiden osalta dokumentointi oli vähintään riittävää. Tekniset dokumentoinnit vaativat käyttäjältään kuitenkin perustietoja ollakseen käyttökelpoisia. Elektroniikkakomponenttien dokumentoinnit vaativat perustietoja sähköopista ja kykyä lukea kytkentäkaavioita. Nämä perustiedot hankin kirjasta *Electronics For Dummies* (Igoe 2005), joka nimensä mukaisesti sopi hyvin lähtötasoiselle kehittäjälle.

Prototyyppien onnistumisen kannalta olennaiseksi muodostui valmiiden projektien kuvaus. Vaikka minulla oli apunani yksittäisten elektroniikkakomponenttien ja ohjelmointityökalujen kattavat dokumentoinnit, tieto pysyi silti tahmeana. Jotta tieto olisi ollut hyödynnettävissä, minun piti kyetä vielä yhdistelemään elementtejä vaaditulla tavalla. Tähän tarvittiin kehittyneemmän käyttäjän tekemää projektikuvausta, joka soveltuisi käytettäväksi omassa projektissani. Kirjat ja käyttäjien projektidokumentoinnit olivat pääasiallisina tiedonlähteinä etsiessäni esimerkkejä aiemmista vastaavista projekteista. Erityisesti kirjojen lineaarisesti etenevät opastukset olivat minulle hyödyllisiä. Kirjojen esimerkkien uusiokäyttöarvoa laski kuitenkin suuresti alan nopea kehitys, jonka mukana jopa verkon ”virallisten” oppaiden pitäminen tuntuu olevan vaikeata, kirjoista puhumattakaan.

Erikoisena, joukosta poikkeavana tietolähteenä täytyy mainita HRMI-piirilevyn (ks. LIITE 1) suunnitellut käyttäjä-valmistaja. Käyttäjä esittelee sivuillaan piirilevylle kehittelemiään valmiita sovelluksia ja koodeja. Perinteisistä kaupallisista elektroniikkavalmistajista poiketen, käyttäjä-valmistajan tarvetietopohjainen lähestymistapa on selkeästi esillä ja hyödynnettävissä. Käyttäjä-valmistaja oli löytänyt uuden suunnittelutilan ja jakoi sen kaikkien halukkaiden kanssa, korvausta vastaan.

Käytettyjen vapaan lähdekoodin tuotteiden kehityksestä ja ylläpidosta vastaa käyttäjien muodostama yhteisö. Niinpä onkin vain loogista, että raja virallisten ja epävirallisten ohjeiden ja esimerkkien välillä on häilyvä. Järjestelmä nojaa kehittyvään evolutionaariseen malliin, jossa parhaat esimerkit (oletetusti) pääsevät esiin. Tein rajanvedon käyttäjien ja virallisten esimerkkien suhteen siten, että kaikki esimerkit jotka oli esitetty tai linkitetty virallisille sivuille, joissa vain rajatulla määrällä käyttäjiä on muokkausoikeudet, ovat virallisia. Tiedon hyödyntämiseksi englannin kielen taito on ehdoton edellytys, sillä suomenkielisiä tietolähteitä on verkossa todella vähän.

Tehdessäni prototyyppejä, kirjasin ylös kohtaamani ratkaisutiedon haasteet ja seuranneen tiedonhaun lähteet sekä haun hyödyllisyyden (Taulukko 1).

Taulukko 1. Ratkaisutiedon haku.

Lähde	Hyödyllinen ratkaisutieto	Epäonnistunut tiedonhaku, tietoa ei voitu hyödyntää.
Kirjat, e-kirjat, maksulliset oppaat.	6	2
Oma tieto	6	3
Keskustelupalstat, blogit ja Käyttäjien tekemä dokumentointi.	12	4
Kaupallisten, suljetun lähdekoodin tuotteiden dokumentointi ja valmistajien jakama tieto.	6	1
Avoimen lähdekoodin tuotteiden dokumentointi ja avoimen tietoyhteisön ”viralliset ”esimerkit.	10	5

6 MUUTTUVA INNOVAATIOYMPÄRISTÖ

Yleisesti lainatun Roy Rothwellin (1994, 7–13) mallin mukaisesti innovaatiot jakautuvat viiteen sukupolveen, jotka kuvaavat aikansa kehitystä ja ympäristöä. Ennen viidettä sukupolvea käyttäjiä on pidetty lähinnä tarve- ja käyttäjätiedon lähteinä. Vasta Rothwellin muotoilemassa viidennen sukupolven innovaatiomallissa mainitaan edelläkävijäkäyttäjien mahdollinen rooli prosessissa. Yritysten tärkeimpiä haasteita

viidennen sukupolven innovaatiomallin saavuttamisessa ovat tiedon jakaminen työntekijöiden kesken, tehokkaiden ryhmätyökalujen käyttö, prosessin nopeuttaminen ja prosessin eri osien rinnakkainen edistäminen (Orlikowski 1993, 363–366; Davenport 1997, 7; Rothwell 1994, 12—13, 24–25). Tavoitteen saavuttamista rajoittavat yrityksen työntekijöiden määrä, työntekijävoiman laatu, tiedon kasaantuminen harvojen saataville sekä kannustimet, jotka laskevat motivaatiota jakaa tietoa oman yksikön ulkopuolelle (Lee & Cole 2003, 24–25).

Yritysten kamppaillessa näiden haasteiden kanssa, innovaatiokentälle on ilmestynyt uusi pelaaja, joka on jo muuttanut pelin säännöt. Avoimen lähdekoodin ideologialla ja sovelluksilla on ollut suunnaton vaikutus tietotekniikka-alaan hyvin lyhyessä ajassa. Kyseessä on ilmiö, joka on muokannut henkisiä arvoja, ansaintamalleja ja uusien innovaatioiden kehitystä. Avoin lähdekoodi tarkoittaa ohjelmia, jotka täyttävät Open Source Initiativen määrittelemät vaatimukset[6]. Ensimmäinen suuri muutos, joka avoimen lähdekoodin liikkeellä on ollut käyttäjien mahdollisuuksiin innovoida, on taloudellinen vapaus. Prototyyppien kehittämisessä käyttämäni ohjelmat olivat kaikki avoimen lähdekoodin sovelluksia, enkä kokenut tarvetta kaupallisille vastikkeille. Käyttäjä ei ole enää sidottu kaupallisiin vaihtoehtoihin ohjelmistojen tai elektroniikan valinnoissa[7]. Tällä on suora vaikutus niiden ihmisten määrään, joilla on mahdollisuus innovoida. Kun laadukkaiden suunnittelu- ja innovaatioresurssien hinta laskee alhaiseksi, niiden levinneisyys kasvaa ja mahdollisuus innovaatioihin levittäytyy entistä tasapuolisemmin ja demokraattisemmin (Hippel 2006, 14).

Internet ja verkkopohjaiset teknologiat ovat mahdollistaneet erikoistuneiden yhteisöjen kommunikoinnin, yhteistoiminnan ja resurssien jakamisen laajemmassa mittakaavassa kuin aiemmin. Tiedon erittäin hajanainen luonne esittää haasteita tiedon tuottamisen prosesseille. Näihin haasteisiin avoimet yhteisöt vastaavat suurella osallistujamäärällä sekä ”monelta monelle” -viestintävälineillä. (Lee & Cole 2003, 2—3.) Tämä kuvaus yhteisöjen toimintatavasta vastaa suoraan aiemmin kuvattuihin haasteisiin, joita yritykset kohtaavat mukautuessaan viidennen sukupolven innovaatiomalliin. Voisi sanoa että kehittämällä työkaluja käyttäjäinnovaatioiden tukemiseksi, avoimet yhteisöt ovat

[6] Luettavissa http://www.opensource.org/docs/osd

[7] Avoin elektroniikka ei kuitenkaan tarkoita ilmaista elektroniikkaa

jopa loikanneet suoraan kuudennen sukupolven innovaatiomalliin. Huomattavaa viidennen sukupolven innovaatiomallissa on käyttäjän rooli, joka on vieläkin useimmissa tapauksissa alisteinen yrityksen tekemälle kehitystyölle. Mielestäni tämä malli ei vastaa enää todellisuutta.

6.1 Käyttäjän muuttuva rooli

Innovaatiosta ja käyttäjän muuttuneesta asemasta on kirjoitettu kasvavassa määrin muutaman viimeisen vuoden aikana. Puhuttaessa käyttäjävetoisesta innovaatiosta pitää tarkentaa, mikä käyttäjän todellinen rooli prosessissa on. Suuressa osassa innovaatiokirjallisuutta käyttäjän roolia tarkastellaan yrityksen näkökulmasta. Käyttäjän kokemusperäistä tarvetietoa arvostetaan tärkeänä osana tuotekehitystä, mutta käyttäjä nähdään harvoin aktiivisena innovoijana.

Käyttäjät kuitenkin todistetusti kehittävät innovaatioita ja muokkaavat tuotteita tarpeitaan vastaaviksi. Miksi näin on? Yksinkertaisesti siitä syystä, että markkinat eivät pysty vastaamaan jokaisen käyttäjän henkilökohtaisiin tarpeisiin (Hippel 2006, 33–34). Yritysmaailmassa on kuitenkin huomattu etuja siitä, että käyttäjälle annetaan mahdollisuus vaikuttaa tuotteiden ja palveluiden sisältöön. Tämä on yksi tapa ratkaista ongelmallinen tarvetiedon siirto käyttäjältä valmistajalle.

”Yritysten ikuisuusongelmana on keksiä, mitä asiakkaat oikeasti haluavat. Käyttäjävetoinen verkkosivusto tarjoaa työkalut, joilla käyttäjät voivat itse luoda palveluita, antaa palautetta ja kertoa ideoistaan.” (Behm 2007.) Tarvetiedon siirtäminen käyttäjätutkimuksen kautta on kallista ja epävarmaa, sillä jokaisella käyttäjällä on oma mielipiteensä, eikä rajallisen otannan tietoihin voi aukottomasti luottaa. (McMeekin ym. 2002, 150–159.)

Pelkän sisällöntuottamisen ja tuotteiden räätälöinnin lisäksi on esimerkkejä siitä, että tuotteiden suunnittelu on kokonaan siirretty käyttäjien vastuulle. Esimerkkinä Hippel mainitsee räätälöityjen puolijohdetuotteiden valmistajat, jotka aiemmin tarjosivat sekä suunnittelu- että valmistuspalveluita asiakkailleen. Käyttäjille suunnattujen suunnittelutyökalujen suosion myötä he menettivät tuotteiden suunnittelutoimintaa työkaluja käyttäville asiakkailleen. Muutos siirsi suunnitteluvastuun ja -vapauden

asiakkaille, jolloin yritykset keskittyivät valmistuskapasiteetin tarjoamiseen, mihin heillä oli hyvät edellytykset suuruuden ekonomian vuoksi[8] (Hippel 2006, 16, 163.) Siirtämällä vastuuta tuotteen tai palvelun suunnittelusta käyttäjälle, valmistajat ovat samalla voimauttaneet käyttäjiä.

6.2 Avoin Elektroniikka

Avoimen lähdekoodin mullistettua ohjelmakehityksen, avoin elektroniikka on tekemässä samaa ihmettä fyysisten tuotteiden kanssa. Koko avoimen elektroniikan käsite on uusi ja perustuu pitkälti juuri avoimen lähdekoodin ideologiaan ja lisenssikäytäntöihin. Avoimella elektroniikalla tarkoitetaan tuotteita joiden tekniset kuvaukset ovat kaikkien saatavilla ja toistettavissa. Kuka tahansa voi halutessaan ladata vaikkapa Arduinon tiedot, teettää levyä Kiinassa ja myydä tuotetta eri nimellä puoleen hintaan. Ei ole vaikeaa huomata, että tämän kaltainen kehitys tulee aiheuttamaan varmasti muutoksia elektroniikka-alalla.

Avoimen lähdekoodin elektroniset kehitysalustat ja uudet tavat hyödyntää olemassa olevia laitteita ovat laskeneet innovaatiokustannuksia käyttäjille. Innovaatioiden saattaminen prototyyppiasteelle ei välttämättä vaadi käyttäjältä suurtakaan taloudellista panostusta. Elektronisten kehitysalustojen hinta on laskenut tasaisesti useimpien komponenttien kanssa.

Elektroniikkavalmistajien keskuudessa on nähtävissä ilmiö, jossa käyttäjille tarjotaan valmistajien laitteissa käytettäviä komponentteja tuotetuen ja kattavan dokumentoinnin kanssa. Nämä komponentit tai tuotteet eivät välttämättä ole avointa elektroniikka, vaan ne voivat olla patenttisuojattuja tuotteita. Niiden tarjoamat mahdollisuudet ovat kuitenkin käyttäjälle hyödyksi. Käyttäjä voi ostaa vaikkapa Nokian puhelimen näyttöä simuloivan kehityspiirilevyn tai Polarin sykemittarissa käytettävän mikroprosessorin, jollaista käytin biometrisen prototyyppini kanssa (LIITE 1). Antamalla käyttäjille mahdollisuuden testata, rakentaa ja leikkiä kehittämillään komponenteilla yritykset voivat saada pienellä vaivalla runsaasti tarvetietoa ja ideoita jatkokehitykselle. Toisaalta tuotteiden suojaaminen viimeiseen asti on usein tuhoon tuomittu yritys, sillä käyttäjien

[8] engl. Economies of scale

kyky ja halu hakkeroida tuotteita johtaisi joka tapauksessa tuotteiden käyttöön, mikäli niille on havaittu uusi kiinnostava käyttötapa.

Kaikki yritykset eivät tosiaan myy tuotteidensa elektronisia osia erikseen tai jaa tietoa niiden ominaisuuksista. Tällaisissa tilanteissa käyttäjien muodostamat yhteisöt yleensä nopeasti havaitsevat tuotteiden potentiaalin ja keksivät tavat käyttää tuotteita uusilla tavoilla. Kuuluisin esimerkki tästä lienee Johnny Chung Leen[9] Wii-pelikonsolin ohjainta käyttävät esimerkit, joissa hän rakentaa peliohjaimesta muutamasta komponentista elektronisen piirtotaulun. Huolimatta siitä miten yhtiöt ovat tarkoittaneet tuotteitansa käytettävän, verkon avulla kukoistava hakkerikulttuuri varmistaa uusien ideoiden leviämisen. Valmiiden tuotteiden ja uuden suunnittelutilan ominaisuuksia hyödyntäviä komponentteja ilmestyy myytäväksi uuden käyttötavan ilmettyä. Näistä esimerkkeinä voisi mainita vaikkapa Wii-pelikonsolin Nunchuck-ohjaimen Arduino-kehitysalustaan yhdistävän adapterin tai iPod-soittimelle tehdyn piirilevysovittimen. Edelläkävijäkäyttäjät luovat näin uuden suunnittelutilan lisäksi kysynnän uusille tuotteille, jolloin käyttäjät saattavat muodostaa omia tuotantoketjujaan kysynnän mukaan. Baldwin, Hienerth ja Hippel kuvailivat prosessin jakamalla käyttäjät kolmeen kategoriaan. Käyttäjä-innovaattorit, käyttäjä-ostajat ja käyttäjä-valmistajat[10] (Baldwin, Hienerth & Hippel 2006, 2, 18). Koska hyvin pieni osa käyttäjistä on varsinaisia edelläkävijäkäyttäjiä tai käyttäjä-innovaattoreita, hyötyvät muut heidän halustaan tuotteistaa innovaatioitaan tai jakaa tietoansa. Käyttäjä-valmistajien tekemien tuotteiden olemassaolo mahdollistaa matalamman ratkaisutietokynnyksen toisille käyttäjille ja siten ne ruokkivat vuorostaan uusia innovaatioita. Esimerkkinä biometrisessä prototyypissä käyttämäni käyttäjä-valmistajan tekemä HRMI-piirilevy, joka perustuu Polarin mikroprosessorille, mutta vaatii käyttäjältään matalamman elektronisen tietotason.

Käyttäjävetoisissa innovaatioissa tuotetukea ei välttämättä ole sanan perinteisessä merkityksessä (Hippel 2006, 49). Kaupallisen tuotteen ostanut kuluttaja on tottunut saamaan tuotteen lisäksi lupauksen laadusta. Yleisesti voidaan olettaa sen sisältävän takuun ja tuen, riittävät ohjeet, varaosien toimituksen jne. Käyttäjien kehittämissä ratkaisuissa vastaavanlaista lupausta ei välttämättä ole. Muiden käyttäjien ohjeita

[9] Lisätietoa http://johnnylee.net/projects/wii/

[10] engl. *user-innovators, user-purchasers, user-manufacturers*

noudatetaan omalla harkinnalla ja vastuu on lukijalla. Tämä mahdollistaa tiedon halvemman siirtämisen muille käyttäjille, mutta käyttäjien kehittelemiä ratkaisuita ei voi pitää yhtä luotettavina tai toimintavarmoina kuin kaupallisia. Koko yhteisöllisen kehityksen voima perustuu sen kykyyn kehittää jatkuvasti tuotteitaan ja havaita virheitä.

6.3 Avoin tieto

Koko informaatioympäristöllä tuntuu olevan avautumisen trendi meneillään. Tästä esimerkkinä Aalto-yliopiston ja Helsingin yliopiston FinnONTO hanke, "jossa kehitetään yhdistetyn avoimen tiedon julkaisemisen teknologiaa ja toimintamalleja kansallisella tasolla.- -Avoimen tiedon[11] lähtökohtana on, että tiedon arvo kasvaa sitä avoimesti jakamalla. "(FinnONTO 2.0, lehdistötiedote 26.3.2010.)

Yhteiskunnan ja yhteisöjen tasolla on toki taloudellisempaa ratkaista jokin ongelma vain kerran ja jakaa tämä tieto. Hippelin (2006, 77) mukaan avoimen jaetun tiedon hyöty tulee kuitenkin esille vasta sen diffuusion kautta. Väittäisin siksi, että tiedon arvo ei kasva vain jakamalla. Tiedon tulee myös tavoittaa sitä tarvitsevat ja olla helposti uusiokäytettävissä.

Käyttäjien jakamaa tietoa on määrällisesti runsaasti saatavilla ja tiedon arvo uusien ratkaisumallien lähteenä on korvaamaton muihin, "jäykempiin " tietolähteisiin verrattuna. Löysin tekemiini prototyyppeihin valmiita osittaisia ratkaisuja (LIITTEET 1-4). Käyttäjien jakaman tiedon uusiokäyttöarvoa laskee kuitenkin muutama tekijä. Käyttäjien jakamaa tietoa on runsaasti saatavilla lähinnä verkosta. Kuten kaiken muunkin tiedon etsimisessä, kyky löytää tarvittu tieto muiden joukosta saattaa osoittautua vaikeaksi. Silkka tiedon määrä ja verkon sirpaleinen luonne tekevät oikean tiedon löytämisestä haasteellista. Löysin runsaasti tarvitsemaani tietoa juuri käyttäjien tekemistä projekteista (taulukko 1), mutta sen löytäminen vei suhteessa enemmän aikaa kuin muiden lähteiden käyttäminen.

[11] engl. open data

Yhtenäisten dokumentointitapojen puuttuminen selittää lähes täysin epäonnistuneet tiedonhaut käyttäjien dokumentoimista projekteista (taulukko1). Vaikka käyttäjät olisivat halukkaita jakamaan innovaatioitansa ja tieto olisi helposti halukkaiden saatavilla, niiden uusiokäyttöarvoa laskee huonosti esitetty tai puuttuva tieto. Yhtenäinen dokumentointimalli, joka sisältäisi lineaarisesti etenevän kuvauksen, kuvia eri vaiheista ja täydelliset versio-, koodi- ja komponenttilistaukset, edistäisivät tiedon diffuusiota.

6.4 Nopean muutoksen vaikutus

Tietotekniikan kanssa nopeaa kehitystä ja innovaatioiden lyhyttä elinikää voidaan pitää itsestään selvyytenä. Varsinkin käyttäjien muodostamat aktiiviset yhteisöt tuottavat kiihtyvällä tahdilla innovaatioita ja uudistuksia. Tämä nopea tahti lyhentää ohjeiden ja dokumentointien hyödyllistä käyttöikää. Projektikuvaus, jossa käytetään vanhentuneita versioita koodeista, vanhoja komponentteja tai muita vanhentuneita resursseja saattaa olla melko hyödytön käyttäjälle, joka etsii suoraa ratkaisua. Myös kirjojen lineaarinen ja selkeä ohjeistus kärsii nopeasta muutoksesta.

Voidakseni hyödyntää *Making Things Talk* – kirjan (Tom Igoe, 2007) esimerkkejä, jouduin etsimään komponentteja, jotka olivat jo selkeästi vanhentuneita. Kirja on julkaistu vuonna 2007 ja kirjan projekteissa käytetyistä Xbee-radiomoduuleista on julkaistu sen jälkeen ainakin 3 uutta versiota[12]. Uudet versiot eivät ole yhteensopivia vanhempien mallien kanssa. Projektikuvauksien arvo tiedon hakijalle saattaa laskea hyvin nopeasti kokoonpanon osien ikääntyessä. Jouduin toteamaan monen projektikuvauksen ja jopa virallisten esimerkkien kohdalla, etteivät ne enää olleet yhteensopivia uusien tuoteversioiden kanssa.

7 YHTEENVETO

Käynnissä on muutos, jonka syyt ovat monitahoiset. Muutoksen juuret ovat luultavasti osaltaan siinä mediassa, joka on avoimen muotonsa kautta myös muotoillut viestiään, eli internetissä. Uusi viestintäkanava on mahdollistanut reaaliaikaisen monelta–monelle-

[12] http://arduino.cc/en/Main/ArduinoXbeeShield, tarkistettu 6.4.2010

viestinnän ja uudenlaisten yhteisöjen syntymisen. Miten nämä ja muut ympäristömme muutokset ovat vaikuttaneet käyttäjien kykyyn tehdä innovaatioita? Nähdäkseni muutoksella on ollut käyttäjiä voimauttava ja innovaatioita tukeva vaikutus.

7.1 Käyttäjäinnovaatioita tukevat muutokset

Avoin lähdekoodi on varmasti eräs merkittävimpiä niistä tekijöistä, mitkä mahdollistivat prototyyppieni tekemisen. Ohjelmistot, kehitysympäristöt ja yhteisöjen tekemä pioneerityö olivat edellytyksiä onnistumiselleni. Avoimen lähdekoodin tuotteet, yhteisöllisyys ja ideologia ovat vahvasti käyttäjäkeskeisiä. Koko muutoksen kantava voima on ollut yksilön halu kehittää uutta, itselleen ja muille. Elämää ja innovaatioita on ollut toki ennen avointa lähdekoodiakin, mutta on päivänselvää että ilmaisuus, avoin jakaminen ja yhteisöllisyys ovat lisänneet käyttäjän työkaluvarastoa merkittävällä tavalla. Miten avoin lähdekoodi oikeastaan auttoi minua projekteissani? Se tarjosi minulle mahdollisuuden käyttää ilmaisia ja helposti saatavilla olevia kehitystyökaluja, edelläkävijäkäyttäjien löytämiä suunnittelutiloja ja koko yhteisön tietomäärää hyväkseni. Kaikki nämä tekijät tukivat minua edullisen innovaatiotilan luomisessa. Työkaluissa oli matala oppimiskynnys, jolloin niiden voidaan katsoa soveltuvan muillekin kuin edelläkävijäkäyttäjille.

Avoimen lähdekoodin ideologia on levinnyt myös elektroniikkaan. Elektronisista kehitystyökaluista, joita ennen pidettiin insinöörien tai muiden asiaan ”perehtyneiden” ihmisten työkaluina, on tullut suuren ihmisjoukon huvitus. Avoimet ja helppokäyttöiset kehitysalustat tarjoavat aivan uudenlaisia mahdollisuuksia käyttäjien innovaatioihin. Käyttäjät ovat aina tehneet innovaatioita elinympäristössään. Niinpä tuntuu luonnolliselta, että käyttäjät jotka elävät elektroniikan keskellä, haluavat myös kehittää sen avulla omia tuotteitaan. Tekniikan monimutkaistuessa tämä on jäänyt harrastuneimpien käyttäjien huviksi.

Näkisin, että uudet kehitystyökalut ovat nyt laskeneet ratkaisutiedon tarvetta, jolloin yhä suurempi joukko käyttäjiä uskaltautuu innovoimaan elektroniikan kanssa. Avoimet elektroniset kehitystyökalut olivat ehdottoman tärkeitä onnistumiseni kannalta. Niiden avulla kykenin toimimaan alueella, josta minulla ei ollut juuri mitään tietoja. Kaikki käyttäjille suunnatut kehitystyökalut, olivat ne sitten digitaalisia tai elektronisia,

laskevat ratkaisutiedon tarvetta ja tekevät tiedosta vähemmän tahmeaa. Tekemistäni prototyypeistä käy ilmi, että lähes jokaisella on mahdollisuus kehittää omia tuotteitaan niin halutessaan.

Avoin muutos on käyttäjävetoista. Edelläkävijäkäyttäjät ja hakkerit mahdollistavat uusien suunnittelutilojen syntymisen. Käyttäjät, jotka kehittävät innovaatioitaan, jakavat tietonsa ja tämä avoin tiedon jakaminen mahdollistaa innovaatioiden diffuusion. Ilman muita käyttäjiä, minun prototyyppini eivät olisi olleet mahdollisia. Käyttäjät ovat jossain mielessä voimauttaneet itse itsensä, ottamalla uudet työkalut ja mahdollisuudet omikseen ja jakamalla tietonsa muiden kanssa. Uusien viestintämahdollisuuksien myötä yhteisöllinen kehittäminen on osoittautunut tehokkaaksi tavaksi kehittää uusia tuotteita ja palveluita.

”Tehokasta innovointia ei saada aikaan enää keskitetyn hallinnoinnin eikä myöskään suuren rahan avulla” (Koivisto 2008, 3). Valmistajien tarjoamat räätälöidyt tuotteet ja uudenlainen suhtautuminen käyttäjien haluun muokata tuotteita, ovat myös osaltaan voimauttaneet käyttäjiä. Tämä on kuitenkin vain askel siihen suuntaan, johon käyttäjien etujoukko on jo rynnistänyt. Avoimet muutokset ja käyttäjien halu kehittää omia tuotteitaan ovat muuttaneet monien yritysten toimialaa. Selviytyäkseen monien yritysten pitää ymmärtää tätä muutosta ja muuttaa toimintamalliaan.

Prototyyppien valmistuttua etsin vastaavia kokoonpanoja internetistä, enkä ollut kovinkaan yllättynyt löytäessäni niitä. Osaltaan tämä kertoo innovaatiomahdollisuuksien laajuudesta ja innovaatioita tekevien käyttäjien suuresta määrästä. Toisaalta löydöt saivat minut arvioimaan prototyyppieni omaperäisyyttä ja yhdenmukaisilla työkaluilla tapahtuvan kehityksen luonnetta. Maailmassa on ainakin 130 000 Arduinon käyttäjää. Ehkä kysymys onkin, kuinka voisin olla löytämättä täysin tai lähes vastaavia projekteja. Kysymys olikin innovaatioideni subjektiivisesta uutuudesta. Kuten työni alussa totesin, uutuus on kokemusperäinen arvo, ei objektiivinen ajan mittari.

7.2 Parannusehdotuksia

Avoimen lähdekoodin projektit ovat yleisesti hyvin dokumentoituja teoksia. Sen sijaan yksittäisten käyttäjien tekemien projektien dokumentoinneissa on valtavasti eroja.

Vaikka tekniselle dokumentoinnille olisikin yhteisesti sovittu muoto, kaikkien käyttäjien ei voi olettaa käyttävän sitä. Siinä missä avoimen tiedon runsauden mahdollistaa käyttäjien suuri määrä ja monipuolinen viestintäalusta, samat tekijät rajoittavat tiedon saavutettavuutta ja käyttöarvoa.

Tarve yhdenmukaiselle ja selkeälle dokumentoinnille on tiedostettu ja esimerkiksi Arduinon osalta puutetta on yritetty korvata Fritzing-ohjelmalla. Minä näkisin hyödyllisemmäksi dokumentointiominaisuuden lisäämisen suoraan kehitysympäristöön. Tällä tavoin käyttäjän ei tarvitsisi huolehtia kehitysympäristön, käytettyjen moduulien ja kirjastojen versioista tai yhteensopivuudesta. Ladattava dokumentointitiedosto yhdessä kehitysympäristön kanssa voisi automatisoida tällaiset toimenpiteet käyttäjän puolesta. Eräs vaikeammin ratkaistava ongelma on projektien käyttökontekstin ja toimintalogiikan selkeä kuvaaminen. Ohjelmoinnin ja elektronisten komponenttien syy – seuraus-suhteet voidaan kerätä ja esittää automatisoidusti, mutta esimerkiksi käyttökontekstin kuvaus jää yhä käyttäjän vastuulle.

Elektronisten komponenttien runsaus asettaa muuttujia, joita ei voida täysin ennakoida työkalujen suunnittelussa. Lähes yhdenmukaiset komponentit voivat poiketa joiltain ominaisuuksiltaan ratkaisevasti, jolloin niiden käyttö korvaavina osina projektissa ei onnistu ilman uutta ratkaisutietoa. Modulaarisesti lisättävät elektroniset laajennukset, esimerkiksi Arduinon Kilvet, auttavat tilannetta, mutta kattavat vain hyvin pienen osan innovaatioiden tarpeista.

Käyttäjiltä vaaditun ratkaisutiedon määrä on laskenut, mutta ei kadonnut. Edullisen innovaatiotilan luomiseen ei kokemusteni mukaan riitä pelkkä työkalujen ja tiedon saatavuus. Käyttämäni kehitystyökalut eivät ratkaisseet monia kohtaamiani ongelmia tai opastaneet prosessissa haluttuun suuntaan. Työkaluista puuttui vielä älykkyys, joka osaisi yhdistellä elementtejä ja ohjata prosessia halutun tavoitteen saavuttamiseksi. Modulaarinen laajennettavuus toimii vain, jos käyttäjällä on itsellään kyky sovittaa uudet ominaisuudet kokonaisuuteen.

7.3 Kuudennen sukupolven innovaatio?

Uudet innovaatiolähteet vaativat uudenlaisia hallintatapoja ja organisaatioita. Markkinatutkimusmetodeja tulee uudistaa käyttäjien kehittämien prototyyppien löytämiseksi ja hyödyntämiseksi. Yrityksillä ei ole käytössään työkaluja innovaatioiden lähteiden analysointiin ja tarvittavan tiedon siirtoon. Niitä tulee kehittää. Eric Von Hippelin jo vuonna 1988 tekemä ehdotus ei ole vieläkään toteutunut suuressa mittakaavassa, mutta merkkejä muutoksesta on jo havaittavissa.

Mikä tulee olemaan käyttäjän rooli tulevaisuuden innovaatioissa? Uskon, että nyt koettu sosiaalisen viestinnän ja yhteisöjen aikaansaama muutos jatkuu entistä tehokkaampien innovaatiotyökalujen muodossa. Uskon, että tulevaisuudessa nähdään älykkäämpiä, jokaisen saavutettavissa olevia työkaluja. Uskon myös, että käyttäjän rooli muuttuu aktiivisemmaksi omien tuotteidensa kehittämisessä, mikäli siihen vain tarjotaan mahdollisuudet.

Ihanteellisessa tilanteessa käyttäjä olisi halutessaan vapautettu ratkaisutiedon etsimisestä. Kuudennen sukupolven innovaatiotyökalut voisivat antaa käyttäjälle mahdollisuuden valita halutut toiminnallisuudet ja rakentaa sovelluksen automatisoidusti käyttäjän valintojen mukaisesti. Elävässä kehitysprosessissa kehitysympäristön älykäs ohjaus voisi ratkaista komponenttien yhteensopivuusongelmat tai valita käytettävät osat ja koodit.

Työkalut poistaisivat tiedon siirron ongelmat yksittäisen käyttäjän innovaatioissa. Tällaisella työkalulla olisi varmasti myös kaupallisia sovellusmahdollisuuksia, sillä markkinat eivät pysty vastaamaan jokaisen käyttäjän henkilökohtaisiin tarpeisiin. Valmistajien näkökulmasta tulevaisuus voisi tarjota vaikkapa modulaarisia rakennuselementtejä, joita käytetään ubiikin ympäristön luomisessa ja niiden käyttöön tarvittavien suunnittelu- ja hallintaohjelmistojen valmistuksessa. Olen kuitenkin varma, että käyttäjän kannalta tulevaisuus on mahdollisuuksia täynnä.

LÄHTEET

Baldwin, Carliss & Hienerth, Christoph & von Hippel, Eric 2006. How user innovations become commercial products: a theoretical investigation and case study. MIT Sloan School of Management Working Paper Number 4572-06.

Behm, Jukka 4.1.2007. Sinä olet tähti. It-viikko. [WWW-dokumentti] <http://www.itviikko.fi/teema/2007/01/04/sina-olet-tahti/20073168/7> (luettu 24.3.2010).

Carr, Wilfred & Kemmis, Stephen 1986. Becoming critical: Education, knowledge and action research. Falmer: London.

Chesbrough, Henry 2006. Open innovation: The New Imperative for Creating and Profiting from Technology. Boston, Massachusetts: Harvard Business School Publishing Corporation.

Davenport, Thomas H. & Prusak, Laurence 1997. Information Ecology: Mastering the Information and Knowledge Environment. New York: Oxford University Press.

DiBona, Chris & Cooper, Danese & Stone, Mark 2005. Open source 2.0: The Continuing Evolution. Sebasopol: O'Reilly Media Inc.

Eskola, Jari & Suoranta, Juha 2005. Johdatus laadulliseen tutkimukseen: 7. painos. Tampere: Vastapaino.

FinnONTO 2.0. Aalto-yliopisto ja Helsingin yliopisto 2010. Lehdistötiedote. < http://www.seco.tkk.fi/events/2010/2010-03-26-linkeddata/finnonto-2-tiedote-2010-03-26.pdf>

Five Generations Of Innovation. [WWW-dokumentti] <http://www.provenmodels.com/575/five-generations-of-innovation/roy-r.-rothwell> (luettu 22.3.2010).

Franke, Nikolaus & von Hippel, Eric 2003. Satisfying Heterogeneous User Needs via Innovation Toolkits: The Case of Apache Security Software. [WWW-dokumentti] Ei julkaisutietoja. Saatavilla: < http://citeseerx.ist.psu.edu/viewdoc/download?doi=10.1.1.13.4172&rep=rep1&type=pdf> (luettu 22.4.2010).

Haglund, Henry 2006. Grid-selvitys. Näkökulma tietojärjestelmien yhteiskäytön edistämiseen. Tilaaja Liikenne- ja viestintäministeriö.

Heikkinen, Hannu L. T. 2007. Ikkunoita tutkimusmetodeihin: metodin valinta ja aineistonkeruu: virikkeitä aloittelevalle tutkijalle. 2.painos. Toim. Aaltola, Juhani & Valli, Raine. Jyväskylä: PS-kustannus.

Igoe, Tom 2010. Adafruit-blogi. [WWW-dokumentti] <http://www.adafruit.com/blog/2010/02/11/130000-arduinos-sold/>

Von Hippel, Eric 1986. Lead Users: An Important Source of Novel Product Concepts. Sloan School of Management, Massachusetts Institute of Technology. Saatavilla <http://web.mit.edu/evhippel/www/papers/Lead%20Users%20Paper%20-1986.pdf>

Von Hippel, Eric 1988. The Sources Of Innovation. New York: Oxford University Press.

Von Hippel, Eric 2006. Democratizing Innovation. Cambridge: The MIT Press.

ISO 13407 standardi. [WWW-dokumentti] <http://fi.wikipedia.org/wiki/ISO_13407> (luettu 22.3.2010).

Karvinen, Kimmo & Karvinen Tero 2009. Sulautet: Opi rakentamaan robotteja ja muita sulautettuja järjestelmiä. Readme.fi Oy.

Keinonen, Turkka & Jääskö, Vesa 2003. Tuotekonseptointi. Helsinki: Teknologiainfo Oy.

Koivisto, Tapio 2009. OpenInno Working paper 2: Kriittisiä havaintoja avoimen

innovoinnin ideasta ja aineksia kestävälle (sustainable) innovaatiotoiminnalle. Tilaaja, VTT. Saatavissa: <http://net.openinno.fi:8080/bin/download/KnowledgeManagement/WebHome/Kriittisihavaintojaavoimestainnovaatiosta.pdf>

Lee, Gwendolyn K. & Cole, Robert E. 2003. From a firm-based to a community-based model of knowledge creation: the case of the linux kernel development. Saatavissa: < http://www.stillhq.com/pdfdb/000501/data.pdf>

McComb, Gordon & Boysen, Earl 2005. Electronics For Dummies. Indiana: Wiley Publishing, Inc.

McMeekin, Andrew & Green, Ken & Tomlinson, Mark & Walsh, Vivien 2002. Innovation By Demand: An interdisciplinary approach to the study of demand and its role in innovation. Manchester: Manchester University Press.

Orlikowski, Wanda J. 1993. Learning from Notes: Organizational Issues in Groupware Implementation," Information Society, 9, 3, July-September, pp. 237-250.

Rogers, M. Everestt 2003 [1962]. Diffusion Of Innovations: 5. painos. New York: Simon & Schuster Adult Publishing Group.

Rothwell, Roy 1994. Towards the fifth-generation innovation process. International marketing review, Vol. 11, No. 1, s. 7.31. Sussex, Englanti: MCB University Press.

Saarinen, Hannu 2009. Käyttäjäinnovaatioiden mahdollisuudet ja ongelmat: tapaustutkimus peliteollisuuden käytännöistä. Helsinki: Helsingin Kauppakorkeakoulu.

Storey, John & Salaman Graeme 2005. Managers of innovation: Insights into Making Innovation Happen. Oxford: BLACKWELL PUBLISHING.

Tekes 2006. Innovaatiotoiminnan vaikutukset: Osaamista, uudistumista, kasvua ja hyvinvointia. Saatavilla <www.tekes.fi/fi/document/42826/innovaatiotoiminnan_vaikutukset_ppt>

Wilson, Scott 2009. Deloitte Review: issue 5. THE OPENMINDED PROFESSOR: An Interview with Eric von Hippel. [WWW-dokumentti] < http://www.deloitte.com/assets/Dcom-UnitedStates/Local%20Assets/Documents/Deloitte%20Review/US_deloitterevie w_EricvonHippel_Jul2009.pdf>

PROTOTYYPEISSÄ KÄYTETYT KIRJALLISET LÄHTEET

O'Sullivan, Dan & Igoe, Tom 2004. Physical computing: sensing and controlling the physical world with computers. Boston: Course Technology.

Reas, Casey & Fry, Ben 2007. Processing: a programming handbook for visual designers and artists. Cambridge: The MIT Press.

McComb, Gordon & Boysen, Earl 2005. Electronics For Dummies. Indiana: Wiley Publishing, Inc.

Igoe, Tom 2007. Making thins talk. Sebasopol: O'Reilly Media Inc.

Greenberg, Ira 2007. Processing: creative coding and computational art. Berkeley: Apress.

LIITTEET

LIITE 1: (1–8)
LIITE 2: (1–13)
LIITE 3: (1–7)
LIITE 4: (1–11)

LIITE 1. BIOMETRINEN PROTOTYYPPI

1 Biometrinen prototyyppi-dokumentointi 1
1.1 Tarvetieto 1
1.2 Mitä syötteitä luetaan? 1
1.3 Mitä syötteellä tehdään? 2
1.4 Miten syke luetaan koneelle? 2
1.5 Tiedon esittäminen 4
2 Käytetyt resurssit 4
2.1 Ratkaisutieto 4
2.2 Ohjelmat 4
2.3 Koodit 4
2.3.1 Sykemittarin lukemiseen tarvittava koodi Processing ohjelmointikielellä 4

1 BIOMETRINEN PROTOTYYPPI-DOKUMENTOINTI

Tämän prototyypin tavoitteena on välittää käyttäjän biometrisiä tietoja koneelle ja esittää ne. Minua kiehtoo ajatus siitä että voisin olla suorassa vaikutuksessa tietokoneeseen kehoni kautta, jopa sellaisten syötteiden välityksellä joihin minulla ei ole tietoista vaikutusta.

1.1 Tarvetieto

Olen käyttänyt sykemittareita ja askelmittareita liikunnan tukena, mutta niiden rajallisuus tiedon käyttömahdollisuuksien suhteen on jäänyt hieman kaivelemaan minua. Nykyään on valmistajien tarjoamia palveluita joilla voi seurata kehitystään ja laatia omia kunto-ohjelmia. Ne vaativat kuitenkin valmistajien omien, uudehkojen mallien käyttöä ja rajoittavat datan käyttöä ohjelmien puitteissa. Minä haluaisin lukea kehollisia/biometrisiä syötteitä ja tehdä niille mitä haluan. Vaikkapa visuaalisen esityksen, pidempiaikaisen tilastoinnin, musiikkia syötteiden mukaan soittavan ohjelman tms.

1.2 Mitä syötteitä luetaan?

Pienen pohtimisen jälkeen päädyin pelkän sykkeen lukemiseen tietokoneelle. Syke on kuvaava syöte joka on myös "rehellinen" siinä mielessä että sen tietoinen ohjailu on vaikeaa. Valintaan vaikutti myös muiden biometristen tietojen lukemisen vaikeus, kalleus ja sensoreiden vähäinen tarjonta[1]. Etsiessäni apua projektin tekoon, löysin vain muutamia käyttäjien tekemiä

vastaavia. Löytämissäni projekteissa sykkeen lukeminen oli ratkaistu melko monimutkaisilla keinoilla, useimmiten hakkeroimalla vanhoja sykemittareita.

1.3 Mitä syötteellä tehdään?

Tavoitteeni tämän prototyypin tiimoilta on lukea syke ja esittää se selkeänä muuttuvana numerona. Jatkokehitys mahdollistaa jälkeenpäin syötteen muuntamisen haluttuihin tarkoituksiin sopivaksi.

1.4 Miten syke luetaan koneelle?

Tutkin sykemittareita ja valmistajien palveluita. Monella valmistajalla on monipuolisia verkkopalveluita harjoitustietokoneiden tietojen analysointiin ja harjoitusohjelmien tekoon. Minä haluan kuitenkin vain lukea sykettä ja tehdä syötteellä mitä haluan.

Haen tietoa monista lähteistä, löydän jonkin verran vastaavia projekteja. Osa vaikuttaa lupaavilta mutta suurimmassa osassa projektin dokumentointi on puutteellinen tai puuttuu kokonaan[2]. Ratkaisut ovat useimmiten selvästi liian monimutkaisia minulle.

Ratkaisuksi löytyy Polar yhtiön Polar Heart Rate Module (Kuva 1)[3] RMCM01.

Kuva 1. Polar Heart Rate Module - RMCM01

Tämä moduuli mahdollistaa Polar sykemittareiden lähettämän signaalin vastaanottamisen. Moduuli vaatii melkoisesti tutkimista ja elektroniikkatyötä toimiakseen, mutta säästää minut sykemittarin elektronisen lähetyksen vastaanoton selvittämiseltä.

Löydän moduulin pohjalta kehitetyn "Polar Heart Rate Monitor Interface-HRMI (kuva 2)[4]" piirilevyn.

Kuva 2. Polar Heart Rate Monitor Interface

Tämä levy muuttaa lähettimen tiedot helpommin luettavaan muotoon ja sisältää myös hyödyllisiä algoritmeja sykkeen keskiarvon mittaamiseksi ja häiriösignaalien poistamiseksi.

HRMI vastaanottaa vanhempien ja joidenkin uudempien sykemittarien lähettimien tietoja. Kantama on heikohko, n. 60 cm, mutta riittää koekäyttöön. Löydän käytetyn Polar Wearlink sykemittarin jonka tulisi sopia levyn kanssa käytettäväksi.

Piirilevyn on suunnitellut Dan Julio ja hänen sivuiltaan löydän selkeän esimerkin Processing-kielellä toteutetusta sykkeen esityksestä. http://danjuliodesigns.com/sparkfun/hrmi.html

HRMIn mukana tulee kattava dokumentointi ja ohjeistus. Onnistun dokumentoinnin avulla lukemaan sykettä ottamalla usb-porttiin kiinnitettyyn piirilevyyn yhteyden terminaaliohjelmalla.[5]

1.5 Tiedon esittäminen

Käytän Dan Julion tekemää Processing esimerkkiä apunani ja onnistun esittämään lähetteen tiedot Processing ohjelmassa, selkeinä muuttuvina numeroina.[6] En kuitenkaan tahdo käyttää valmista koodia sellaisenaan vaan muokkaan sitä mieleisekseni, jolloin esityksestä tulee graafinen muuttuja, sekä vaihtuva sykearvo numeroina esitettynä.[7]

Jatkokehitysideoita:

- Laitteen muuttamista langattomaksi, kannettavaksi versioksi.
- Musiikkisoitin joka reagoi sykkeen muutoksiin.
- Visuaalinen/Audio esitys joka reagoi sykkeen muutoksiin.

2 KÄYTETYT RESURSSIT

2.1 Ratkaisutieto

Lähde	Ratkaisutieto	Epäonnistunut
Kirjat		
Oma tieto/IP	+	
Keskustelupalstat, blogit, Käyttäjien tekemä dokumentointi	++	-
tuotteiden ja yritysten dokumentointi	++	
Avoimen lähdekoodin tuote. Dokumentointi		

2.2 Ohjelmat

Processing
Terminaaliohjelma HyperTerminal

2.3 Koodit

2.3.1 Sykemittarin lukemiseen tarvittava koodi Processing ohjelmointikielellä

```
import processing.serial.*;

/*
HRMI lukija ja sykkeen esitys päivittyvänä graafina&numerona
Alkuperäinen koodi:
Dan Julio
```

danjuliodesigns.com

```
// Muuttujat
Serial port; // Seriaaliportti
byte[] raakaDataArray = new byte[32];
//bittimuotoinen raakadata-array

 int[] palauteArvoArray = new int[3];
// Tähän arrayyn laitamme muutetut vastausarvot, tiedot ovat
// [0] Mode-Average/Raw [1]Palautetun arvon järjestysnumero [2]Syke

int validData = 0;
int CR = 13;
//ASCII muunnettu carriage return

void setup() {
  size(500, 300);
  fill(101,154,167);
  noStroke();

  rect (0,0,width, height);
  port = new Serial(this, "COM6", 9600);
  //toinen vaihtoehto on kutsua portti listasta, Serial.list()[0]
  port.bufferUntil(CR);
  //komennetaan bufferointi kunnes kutsutaan serialEvent()
}


//muutama muuttuja palloa varten
int xpos = 0;
int ypos = 200;
int xspeed=1;
int yspeed=0;

//Määritellään hiukan muuttujia ja arvotaan fontti kaikista koneella olevista.
   String[] fontList = PFont.list();
   int fonttiMaara = fontList.length;
   float arvottuFontti = random(fonttiMaara);
   int intArvottuFontti = int(arvottuFontti);
//   println(fontList);
//   println (intArvottuFontti);
//   println (fonttiMaara);

void draw() {
  // Haetaan yksi sykearvo
  validData = 0;
  port.write('G');
  //hae syke
```

```
port.write('1');
// viimeisin arvo
port.write(CR);
// carriage return
    while (validData == 0) {
//    delay(0); // Odotetaan 1 sekunti tarkistusten välillä
    }
// moodi, järjestysluku ja syke
if ((palauteArvoArray[0] & 0x01) == 0x01)
print("Averaged mode ");
else
print("Raw mode ");

print(palauteArvoArray[1]); print(" "); // järjestysluku
println(palauteArvoArray[2]); // Syke


//Seuraavaksi määritetään tausta ja pallo joka kuvaa sykettä
 // background(0);

//Tässä laatikko sykeluvun taakse
   noStroke();
   fill(101,154,167);
   rect (0,0, 200, 100);


 fill(211,15,19);
 stroke(255,50);
 strokeWeight(3);
  ellipse(xpos, ypos, 5, 5);
  xpos+=xspeed;
  ypos=200-(palauteArvoArray[2]);
  tarkistatormays(xpos, ypos);



 /* tässä on perusmalli kun halutaan pysyä tietyssä fontissa
   PFont myFont;
   myFont = loadFont("AachenBT-Roman-48.vlw");
   smooth();
   fill(255);
   textFont(myFont, 48);
   text("Syke", 50,50);
   text(palauteArvoArray[2], 100,100);  */

 //Tässä käytetään arvottua fonttia
   PFont myFont;
   myFont = createFont((fontList[intArvottuFontti]), 32);
```

```
    smooth();
    fill(255);
    textFont(myFont);
    text("SYKE", 30,30);
    text(palauteArvoArray[2], 100,100);
}



/*
SerialEvent:ä voidaan käyttää usean portin kuunteluun.
tässä on perussyntaksi:

void serialEvent(Serial whichPort) {
  statements
}

The which parameter contains the name of the port where new data
is available, but is only useful when there is more than one serial
connection open and it's necessary to distinguish between the two.
esim.
void serialEvent(Serial p) {
  inString = p.readString();
}
*/

//tarkistus että dataa on
void serialEvent(Serial port) {
if (port.readBytesUntil(CR, raakaDataArray) != 0) {
//asciit numeroiksi
int raakaIndex = 0;
int palauteIndex = 0;
for (int i=0; i<3; i++) palauteArvoArray[i] = 0;
while (raakaDataArray[raakaIndex] != CR) {
if (raakaDataArray[raakaIndex] != ((byte) ' ')) {
palauteArvoArray[palauteIndex] = (palauteArvoArray[palauteIndex]*10) +
(raakaDataArray[raakaIndex] -
((byte) '0'));
} else {
palauteIndex++;
}
raakaIndex++;
}
validData = 1;
}
}

//Tarkistus osuuko laatikko ruudun reunoihin
```

```
void tarkistatormays(int xp, int yp){
  if (xp>width-10 || xp<=0){
    xspeed*=-1;
  }
  if (yp>=width-10 || yp<=0){
    yspeed*=-1;
  }
}
```

---

[1] Useita lähteitä. Keskustelupalstoja ja projekteja sekä verkkokauppojen tarjonnan selvittämistä. +
[2] löydän Googlen haulla 6 vastaavaa projektia (ja monia lähes vastaavia), kaikki eri lähteistä. Puutteellinen dokumentointi ja projektikuvaus. -
[3] yritys tukee käyttäjien innovointia myymällä erillistä komponenttia +
[4] Tuote on uusi, enkä löydä muita sitä käyttäviä projekteja. Lead User käyttäjän luoma piirilevy. Joku muu on jo ratkaissut ratkaisutiedot puolestani. (http://www.coolcomponents.co.uk/catalog/product_info.php?products_id=279). +
[5] Dokumentointi, käyttäjä-valmistajan tekemä. +
[6] Esimerkki User-Manufacturer toimijalta, verrattavissa dokumentointiin. +
[7] Käytän esityksen tekemiseen omia aiempia Processing koodejani. +

LIITE 2. KOTIVAHTI PROTOTYYPPI

1. kotivahti prototyyppi-dokumentointi 1
1.1. Tarvetieto 1
1.2. Miten ratkaisen etäohjauksen? 2
1.3. Miten järjestelmä rakentuisi? 2
1.4. Ethernet yhteyden saaminen 3
1.5. Radioyhteys 5
1.6. Yhdistetty, toimiva järjestelmä. 6
2. Käytetyt resurssit 8
2.1. Elektroniikka 8
2.2. Ratkaisutieto 9
2.3. Ohjelmat 9
2.4. Koodit 9
2.4.1. Ethernet yhteys ja hallintapaneeli Processing ohjelmointikielellä 9
2.5. Xbee-radioiden yhteys Arduino ohjelmointikielellä 11

1. KOTIVAHTI PROTOTYYPPI-DOKUMENTOINTI

Kun lähdet kotoasi pidemmäksi ajaksi matkoille tms. jätä jokin valo palamaan tai pyydä jotain tuttuasi sytyttelemään valoja samalla kun kastelee kukkasi. Ideana tässä lienee huijata tarkkaavaisia murtomiehiä. Kuulostaa uskomattomalta että ammattimainen murtoveikko pelkäisi yhtä ainoaa viikon päällä olevaa valoa, eikä minulla ole kukkia kasteltavaksi. Mikä ratkaisuksi?

1.1. Tarvetieto

Tavoitteena on toteuttaa järjestelmä jonka avulla voisin etähallinnoida lähtöjä (valoja, laitteita, radiota yms.). Laitteita tulisi voida kytkeä päälle ja pois päältä tai mahdollisesti antaa ohjelmoinnin sattumanvaraisesti tehdä se puolestani.

Mitä vaatimuksia minulla on prototyypille?

- Kytkeä virta päälle/pois x-laitteisiin
- toimia etäohjattuna
- osoittaa toiminnan tila, päällä/pois
- mahd. sattumanvarainen toiminto

1.2. Miten ratkaisen etäohjauksen?

Mitä vaihtoehtoja etäohjaukselle on?[1]

- gsm
- puhelinverkko/internet(ethernet)
- radio
- bluetooth
- muita, esim. infrapuna jne.

Vaihtoehdoista vain gsm ja ethernet vaikuttavat mahdollisilta, sillä muut tekniikat ovat kantamaltaan hyvin rajallisia. Saatan tarvita jotain lyhyemmän kantaman tekniikkaa, mikäli käytän useampaa kuin yhtä laitetta.

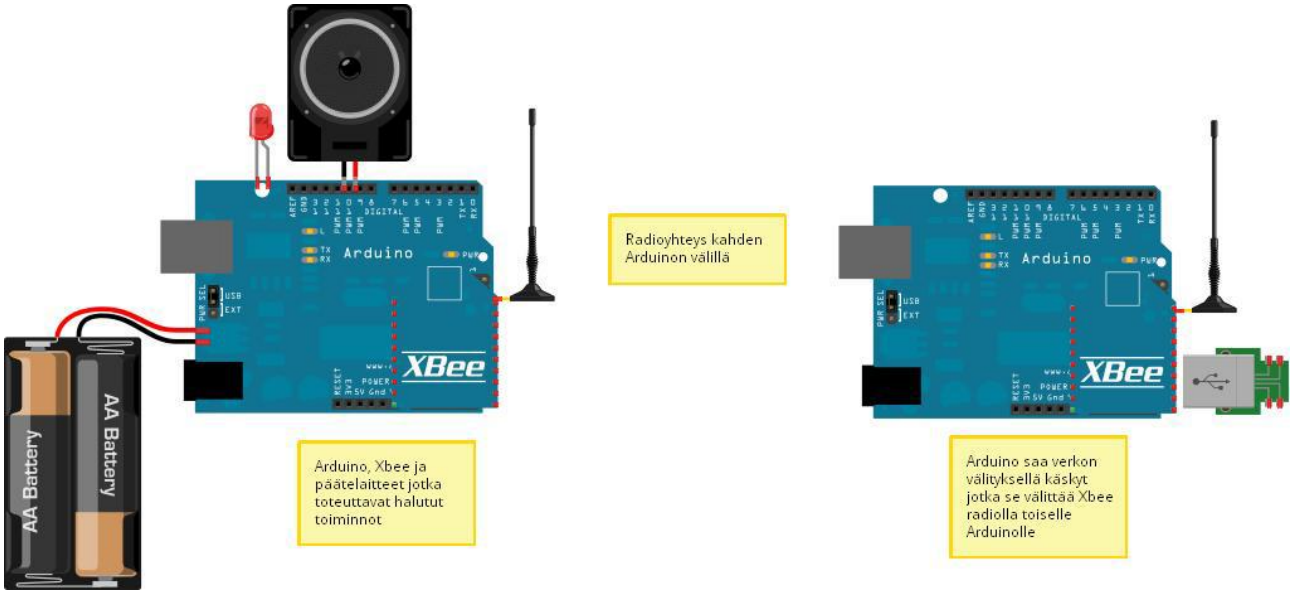
Päädyn käyttämään ethernet yhteyttä, sillä se mahdollistaa verkkokäyttöliittymän rakentamisen ja etähallinnoin mistä tahansa verkon välityksellä. Lisäksi gsm järjestelmä toisi runsaasti lisää kustannuksia. Verkon yli tehtävä ohjaus saattaa mahdollistaa vielä tuntemattomien käyttötapojen ja mahdollisuuksien löytymisen.
Arduinolle on tehty ethernet-shield joka mahdollistaa piirilevyn yhdistämisen suoraan verkkoon.[2]

1.3. Miten järjestelmä rakentuisi?

Tarvitsen yhden laitteen joka on ethernet yhteyden kautta hallittavissa. Jotta järjestelmä ei rajoittuisi vain yhden laitteen lähtöihin, tarvitsen useamman laitteen joissa on erilaisia lähtöjä. Tämä siitä syystä että yhdestä laitteesta toteutetut lähdöt tarvitsisivat pitkät kaapeloinnit. Monen laitteen muodostama verkko saattaa avata myös mahdollisuuksia muihin käyttötarkoituksiin. Tutkin muita yhteystekniikoita ja päädyn radiolähettimien käyttöön monestakin syystä[3].

- Halpa hinta
- laajennettavuus
- Tekniikan (oletettu) helppokäyttöisyys[4]
- Arduinolle valmistettu Xbee-shield mahdollistaa nopean käyttöönoton
- Ei esim. infrapunan rajoitusta suorasta näköyhteydestä

Suunnittelen prototyypin kokoonpanon seuraavanlaiseksi (kuva 1).



Kuva 1. Hahmotelma kotivahtiprototyypistä. Toteutettu Fritzing ohjelmalla.

1.4. Ethernet yhteyden saaminen

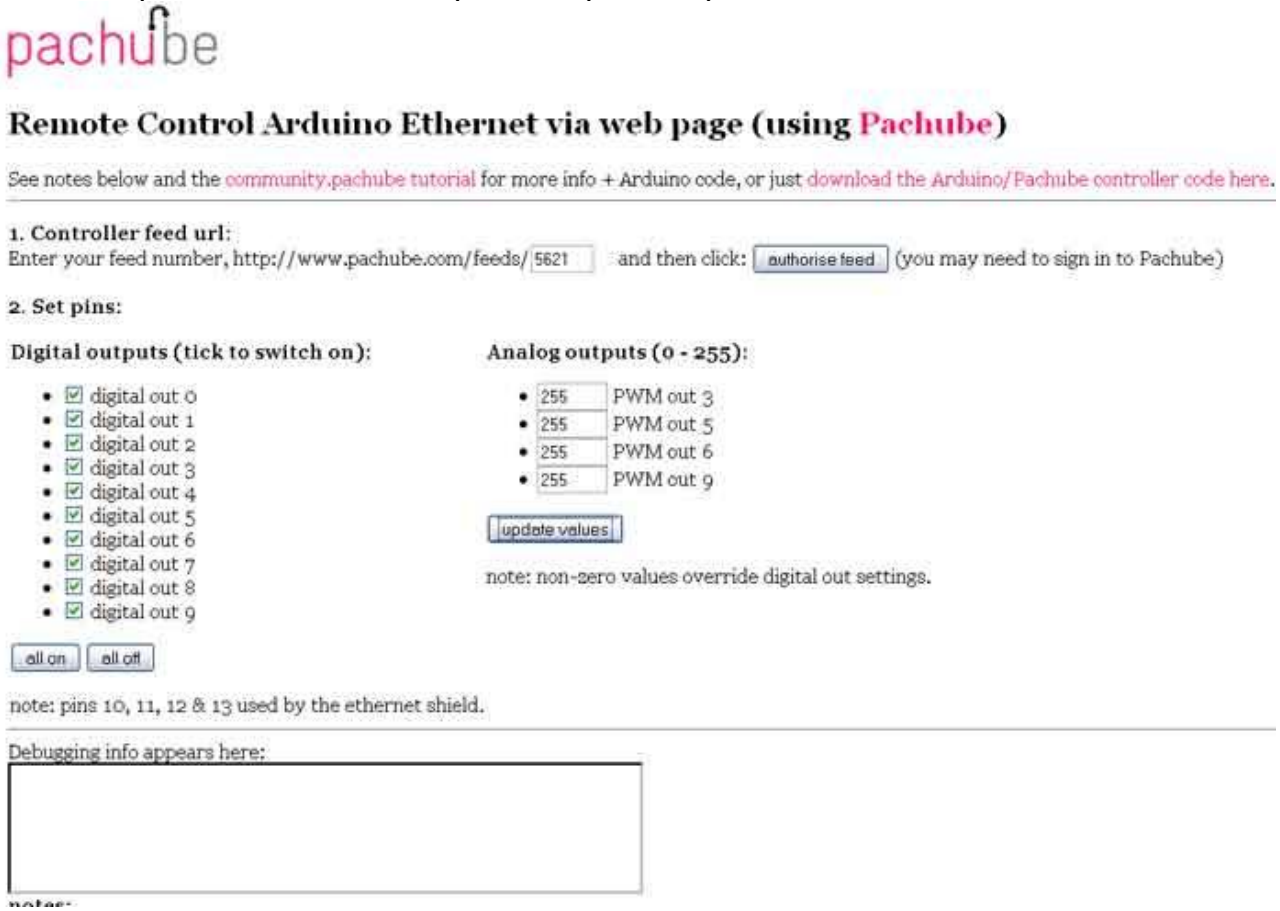
ARDUINO VERKOSSA
Arduinolle löytyy valmis kirjasto Ethernet yhteyttä varten[5]. Fyysiseen yhteyteen vaaditaan Ethernet-liitos, johon käytän valmista ethernet-shield lisäosaa.
Pienen tutkimisen jälkeen ilmenee että yhteyden rakentaminen ei ole kovin yksinkertainen temppu, vaan vaatii melkoisesti säätöjä ja asetuksia.[6]
Löydän Arduinon tukisivustolta tietoa Pachube nimisestä palvelusta joka vaikuttaa helpommalta tavalta yhteyden luomiseen. Päätän kokeilla palvelua sillä se tarjoaa yhteyden lisäksi loistavia mahdollisuuksia syötteiden hyödyntämiseen.
Joudun silti selvittämään Ethernet-shieldin toimintaa. Mac-address ja IP haussa. Arduinon Webserver-kirjaston tutoriaalit ovat kohtalaisia, eivät vain auta raudan asetusten selvittämisessä.

Onnistun lukemaan Analog pin 1-5 ip-osoitteeseen http://10.0.0.4/ Pinneissä ei ole mitään sensoreita joten minun täytyy varmistaa toiminta jollain syötteellä. Käytin tähän Arduinon Ethernet-kirjaston Webserver-esimerkkiä.[7]

Jatkan Pachuben asennusta, vaatii 3 Processing kirjastoa sekä yhden Arduinolle.[8]
Jouduin ottamaan selvää reitittimeni toiminnasta[9], lisäämään 2 kirjastoa Arduinoon, enkä siltikään saa Pachubea toimimaan. Palvelussa ei ole kuin alkeellinen virheenetsintätoiminto, enkä löydä palvelun ohjeistuksesta tai keskustelupalstoilta apua. Kirjoittamaani apupyyntöön ei vastata 2 vrk sisällä.[10]

Kokeilin vielä reitittimenvaihtoa, ilman tulosta. Pachube olisi tarjonnut paljon houkuttelevia ominaisuuksia, mutta prototyyppiprosessin käyttöön sen käyttöönotto on liian hidasta ja vaivalloista. Kyse saattaa olla jostain tapauskohtaisesta ongelmasta jota en onnistunut selvittämään asettamassani aikarajassa, 2 vuorokaudessa.

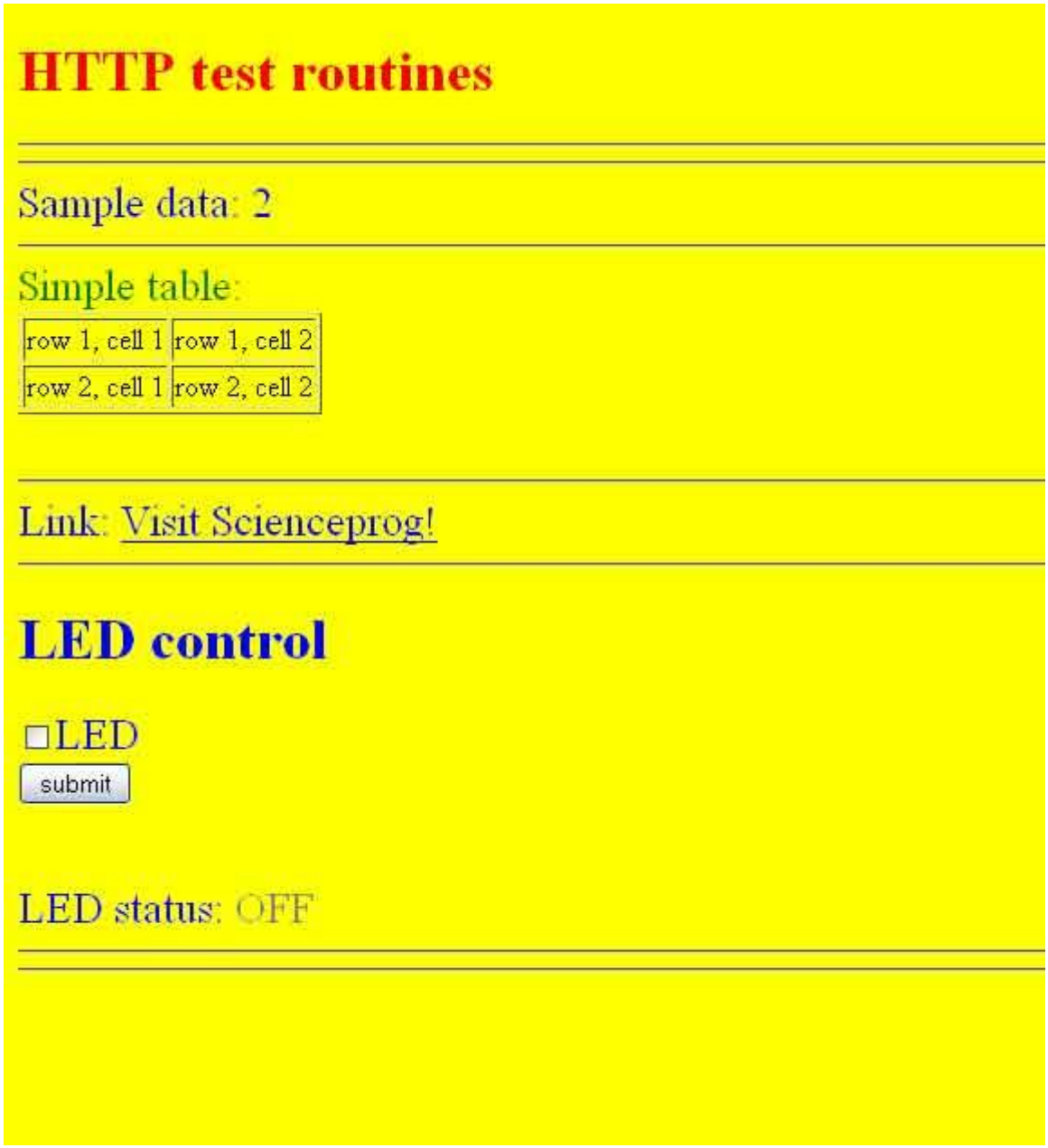
Pachube palvelun etähallintapaneeli (kuva 2).



Kuva 2. Pachube palvelun etäpalvelu.

Toteutan Arduinon kauko-ohjauksen koodinpätkällä jonka löysin Arduino forumin keskustelupalstalta[11]. Tämä koodi ohjaa laitetta ainakin lähiverkossa (reitittimen jakamalla ip-osoitteella) ja toimii selaimessa. Koodilla voidaan kytkeä Arduinon analogisia tai digitaalisia kytkimiä päälle/pois päältä. Ratkaisulla kykenen demonstroimaan etähallittavan järjestelmän toiminnallisuutta, vaikka etähallinta oman lähiverkon ulkopuolelta ei vielä olisikaan mahdollista. Muokkaan mallia omiin tarpeisiini[12].

Alkuperäinen esimerkki hallintapaneelista (kuva 3).

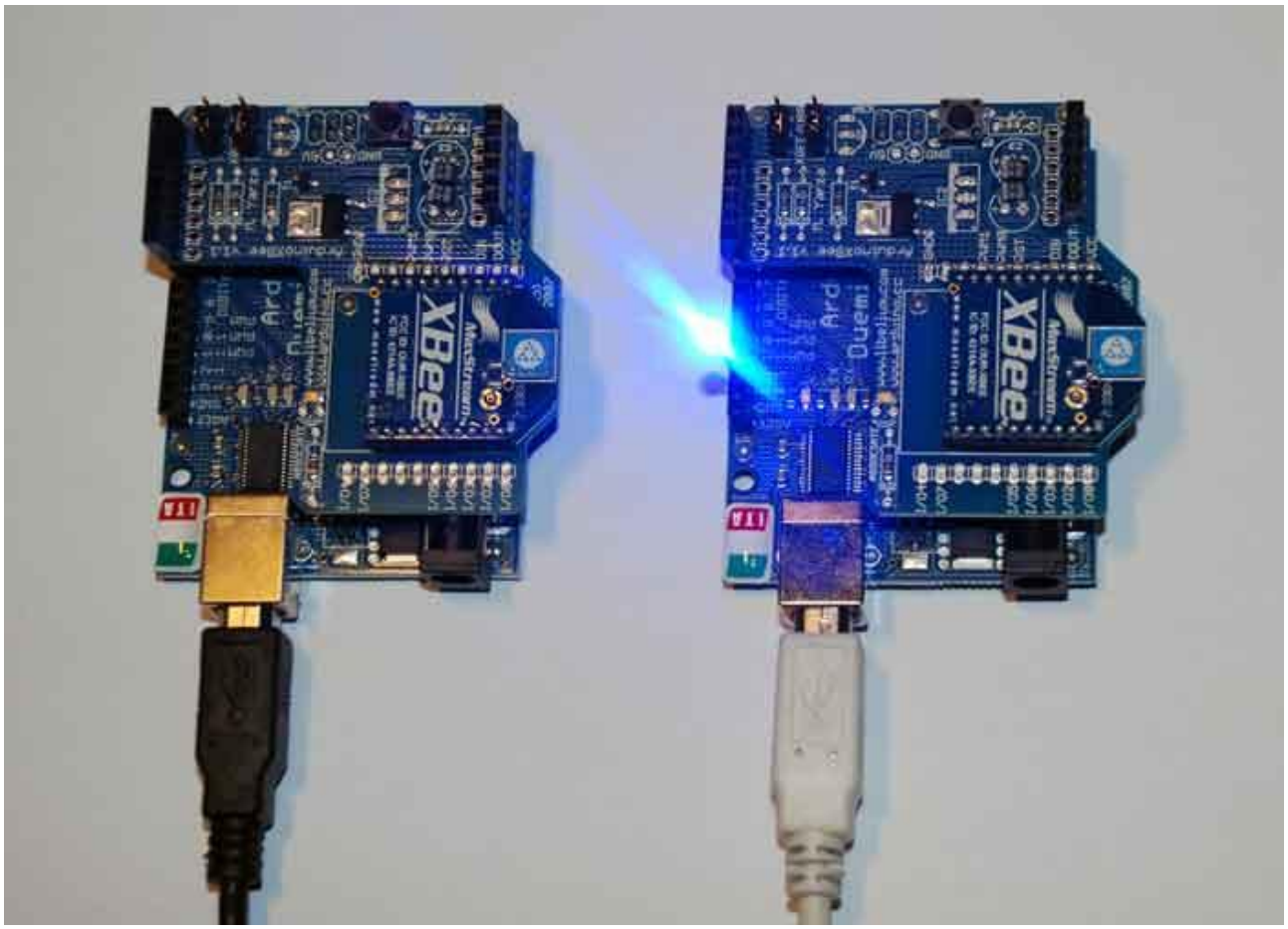


Kuva 3. Arduinon etäkäyttöpaneeli. (sytytä/sammuta LED digitaalisessa kytkimessä nro. 4).

1.5. Radioyhteys

Seuraava haasteeni on saada aikaiseksi radioyhteys kahden Arduinon välille. Kokeilen kahden Xbee radion yhteyttä selkeillä ohjeilla jotka löytyvät Arduinon sivustolta[13]. Radioiksi olen valinnut ensimmäisen sukupolven radiot, sillä suurin osa ohjeistuksista ja esimerkeistä on kirjoitettu niille. Uudemmat versiot eivät ole yhteensopivia vanhojen kanssa ja eroavat toimintalogiikaltaan selkeästi ensimmäisestä sukupolvesta[14].

Sain radiot keskustelemaan perusasetuksilla, toinen radio lähettää käskyn ja toinen suorittaa ohjelmoinnin mukaisesti lähtöjen kytkemisen päälle tai pois päältä (kuva 4).



Kuva 4. Xbee yhteys kahden Arduinon välillä. Tässä esimerkissä viestin välitys ilmenee syttyvänä LED valona.

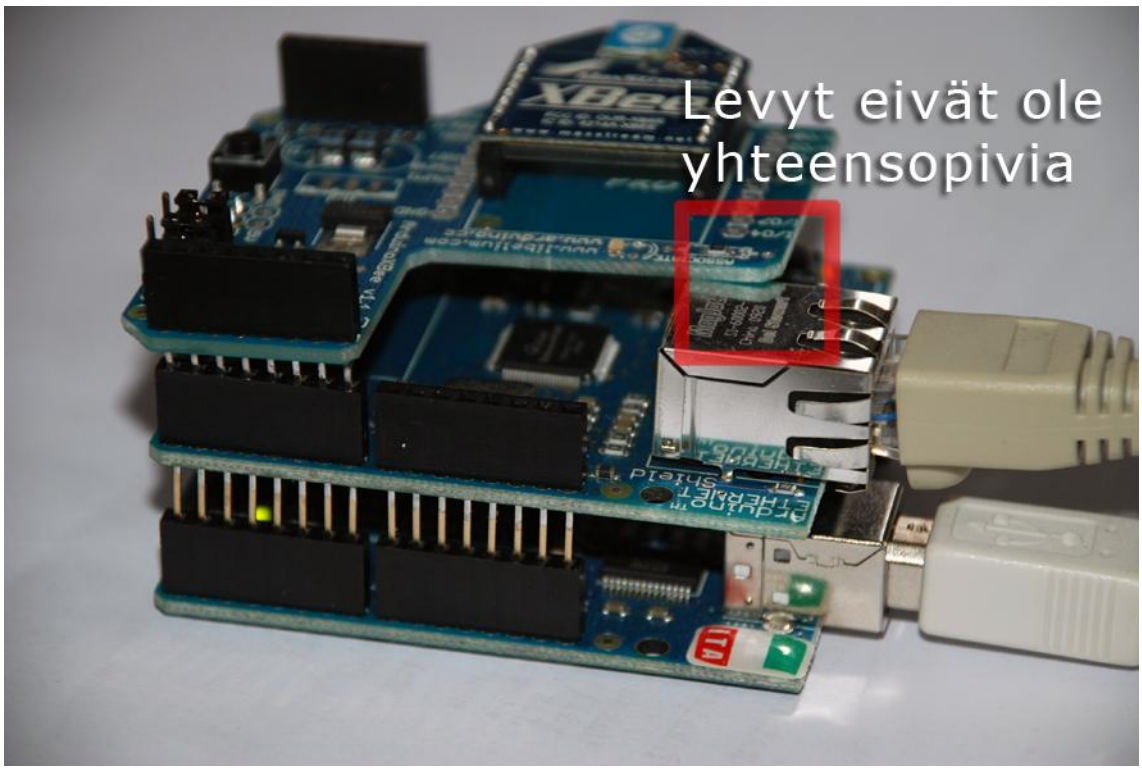
1.6. Yhdistetty, toimiva järjestelmä.

Xbee-radioilla varustetut Arduinot saavat toisiinsa yhteyden ja onnistun saamaan yhteyden ethernet-liittimellä kytkettyyn arduinoon. Nyt minun pitäisi yhdistää ethernet kytkettyyn arduinoon xbee moduuli, jolloin voisin lähettää sillä toiselle xbee:lla arduinolle käskyjä.

Törmään ongelmaan. Ilmeisesti yhteen arduinoon ei voi kasata päällekkäin ethernet- ja xbee-moduulia[15]. Vaikka useimmat arduinon moduulit ovat ”kasattavia ”, eli niitä voi lisätä toistensa päälle, nämä kaksi eivät ole yhteensopivia (kuva 5).

Minun täytyy siis yhdistää xbee-lähetin koekytkentälevyn kautta. Xbeen piikkirima/yhdysnastat ovat tiheämmässä(2mm) kuin tavallisen koekytkentälevyn piikit, joten vaihtoehdoksi jää keksiä jotain muuta tai kolvata kaapelit suoraan radiolähettimeen.[16]

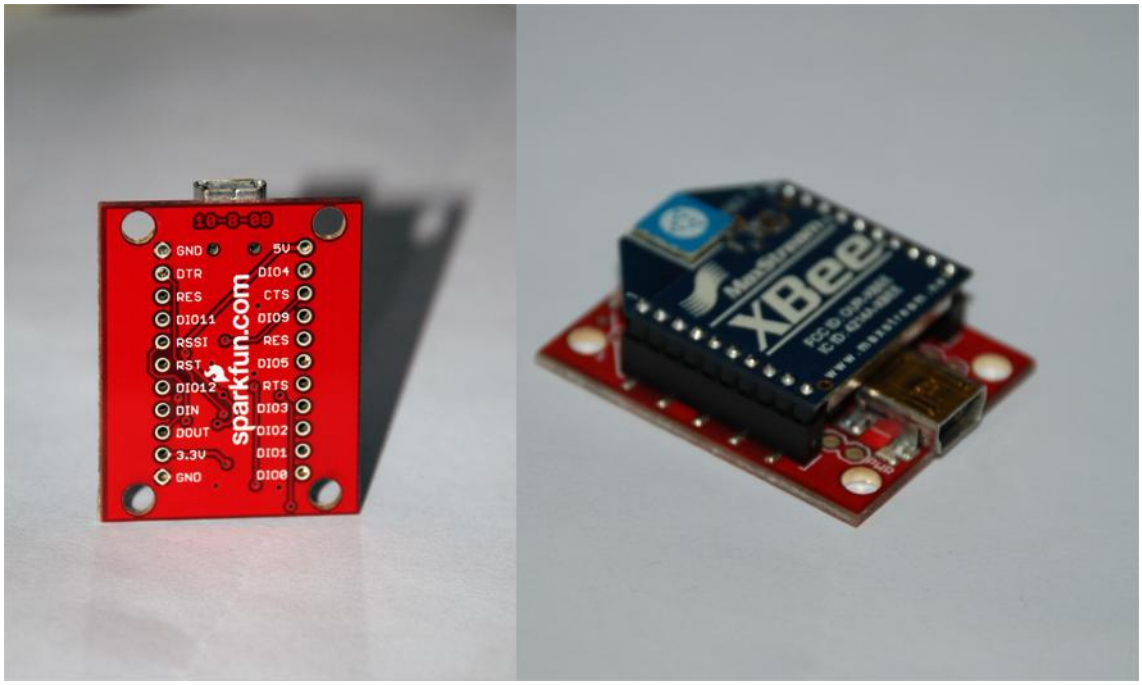
Suljen kolvaamisen pois vaihtoehdoista, sillä se estäisi lähettimen käytön xbee-shieldin kanssa jatkossa.



Kuva 5. Yhteensopimattomat Shieldit

Yritän yhdistää xbee moduulia ja ethernet yhdistettyä arduinoa koekytkentälevylle Xbee-Explorer (USB:stä serialiin) kytkentälaitteen avulla.

Explorerissa on piikkirimoilla paikat pohjassa, mutta toimiva yhteys vaatisi kolvaamista. Exporerin tarkoitus on toimia Xbeen yhdistämiseen ja ohjelmointiin/asetusten asettamiseen tietokoneelta. En tahdo riskeerata Exploreria kolvaamisella, sillä vastaavanlaisen operaation onnistumisesta en löydä viitteitä[17] (kuva 6).



Kuva 6. Xbee Exlorer.

Päätän tyytyä saavutettuihin tuloksiin tältä erää. Tilaan pienen piirilevyn joka on tarkoitettu Xbee-radion yhdistämiseen piirilevyn kautta Arduinoon.

Mitä saavutettiin?

- Sain aikaiseksi selaimessa toimivan hallintapaneelin jolla voidaan ohjata Ethernet yhdistetyn Arduinon lähtöjä.
- Sain yhdistetyksi kaksi Xbee-radioilla toimivaa Arduinoa.
- EN saanut yhdistettyä Ethernet-Arduinoa ja Xbee-Arduinoa.

Jatkokehitysideoita:

- Useiden laitteiden lisääminen ja monen Xbeen käyttö matriisiverkossa.
- Laitteiden ohjaaminen tietokoneelta käyttöliittymän kautta
- Verkkovirtalaitteiden ohjaus

2. KÄYTETYT RESURSSIT

2.1. Elektroniikka

2 x Arduino Duemilanove
2 x Xbee shield
2 x Xbee series 1 radio
1 x Ethernet shield
1 x Xbee Explorer
Koekytkentälevyjä
Hyppykaapeleita
Serial –USB kaapeleita
Ethernet kaapeli

Erilaisia lähtöjä:

Ledejä
Piezo summeri

2.2. Ratkaisutieto

Lähde	Ratkaisutieto	Epäonnistunut
Kirjat	4	2
Oma tieto/IP	1	
Keskustelupalstat, blogit, Käyttäjien tekemä dokumentointi	4	3
tuotteiden ja yritysten dokumentointi	3	1
Avoimen lähdekoodin tuote. Dokumentointi	5	2

2.3. Ohjelmat

Processing
Arduino
Fritzing
Terminaaliohjelma HyperTerminal

2.4. Koodit

2.4.1. Ethernet yhteys ja hallintapaneeli Processing ohjelmointikielellä

```
#include <WString.h>
#include <Ethernet.h>
/*
Yksinkertainen Ethernet testi
alkuperäinen koodi:
      By Minde
            http://www.sciencprog.com/

Vaatii:
            * Arduino Duemilanove
      * Arduino Ethernet shield
      *LED kiinnitetty GND ja digital pin 4:ään resistorin kautta
*/

byte mac[] = { 0xDE, 0xED, 0xEE, 0xEE, 0xFE, 0xED };
byte ip[] = { 10, 0, 0, 4 };
byte gateway[] = { 10, 0, 0, 2 };
```

```
byte subnet[] = { 255, 255, 255, 0 };
Server server(80);
byte sampledata=50;
int ledPin = 4;  // LED pin
String readString = String(30);
boolean LEDON = false; //LED tila
void setup(){
//aloitetaan Ethernet
  Ethernet.begin(mac, ip, gateway, subnet);
//asetetaan pin 4 lähdöksi
  pinMode(ledPin, OUTPUT);
  Serial.begin(9600);
}
void loop(){
// Yhdistetään Client
Client client = server.available();
  if (client) {
    while (client.connected()) {
   if (client.available()) {
    char c = client.read();
    //luetaan char kerrallaan HTTP request
    if (readString.length() < 30)
      {
       //tallennetaan luvut/kirjaimet string:iin
       readString.append(c);
      }
       //printataan serial.port:iin
       Serial.print(c);
      //jos serial.request on loppu
       if (c == '\n') {
         //tarkistetaan pitääkö ledin olla päällä
        if(readString.contains("L=1"))
          {
           //ledin pitää olla päällä
          digitalWrite(ledPin, HIGH);    // set the LED on
           LEDON = true;
          }else{
           //ledin pitää olla pois päältä
          digitalWrite(ledPin, LOW);    // set the LED OFF
           LEDON = false;
          }
         // HTML
         client.println("HTTP/1.1 200 OK");
         client.println("Content-Type: text/html");
         client.println();
         client.print("<body style=background-color:yellow>");
         client.println("<font color='red'><h1>HTTP test
routines</font></h1>");
```

```
        client.println("<hr />");
        client.println("<hr />");
        client.println("<font color='blue' size='5'>Sample data: ");
        client.print(sampledata);/
        client.println("<br />");
        client.println("<hr />");
        client.println("<font color='green'>Simple table: ");
        client.println("<br />");
        client.println("<table border=1><tr><td>row 1, cell 1</td><td>row
1, cell 2</td></tr>");
        client.println("<tr><td>row 2, cell 1</td><td>row 2, cell
2</td></tr></table>");
        client.println("<br />");
        client.println("<hr />");
        client.println("<h1>LED control</h1>");
        client.println("<form method=get name=LED><input type=checkbox
name=L value=1>LED<br><input type=submit value=submit></form>");
        client.println("<br />");
        client.print("<font size='5'>LED status: ");
        if (LEDON)
           client.println("<font color='green' size='5'>ON");
        else
           client.println("<font color='grey' size='5'>OFF");
        client.println("<hr />");
        client.println("<hr />");
        client.println("</body></html>");
        readString="";
        client.stop();
          }
        }
      }
    }
 }
```

2.5. Xbee-radioiden yhteys Arduino ohjelmointikielellä

```
/*
Xbee RADIO 1

Alkuperäinen koodi:
created 2006
 by David A. Mellis
 modified 14 Apr 2009
```

```
 by Tom Igoe and Scott Fitzgerald
 http://www.arduino.cc/en/Tutorial/PhysicalPixel

Runsaasti lyhennetty ja muokattu tämän projektin tarpeisiin
 */

const int ledPin = 13; // the pin that the LED is attached to
int incomingByte;      // a variable to read incoming serial data into

void setup() {
  // initialize serial communication:
  Serial.begin(9600);
  // initialize the LED pin as an output:
  pinMode(ledPin, OUTPUT);
}

void loop() {
  // see if there's incoming serial data:
  if (Serial.available() > 0) {
    // read the oldest byte in the serial buffer:
    incomingByte = Serial.read();
    // if it's a capital H (ASCII 72), turn on the LED:
    if (incomingByte == 'H') {
      digitalWrite(ledPin, HIGH);
    }
    // if it's an L (ASCII 76) turn off the LED:
    if (incomingByte == 'L') {
      digitalWrite(ledPin, LOW);
    }
  }
}


/*
Xbee RADIO 2
Itsekirjoitettu
*/
void setup()
{
  Serial.begin(9600);
}

void loop()
{
  Serial.print('H');
  delay(1000);
  Serial.print('L');
  delay(1000);
```

}

[1] Monia lähteitä. Arduino keskustelualue, blogeja, Making thing talk kirja, Physicall Computing kirja. +
[2] http://www.arduino.cc/en/Main/ArduinoEthernetShield. +
[3] Monia lähteitä, pääasiassa tehtyjen elektroniikkaprojektien kuvauksia, blogeja +
[4] Making Things Talk, s.192. Monia verkkolähteitä +
[5] http://arduino.cc/en/Reference/Ethernet Valmis kirjasto ja ohjeistus, Processing A Programming handbook kirja +
[6] Monia lähteitä. Ongelmaksi muodostuu laitekohtaisten osoiteasetusten selvittäminen. Kokeilen myös kirjalähteiden esimerkkejä ja neuvoja tuloksetta.(Making things Talk, Processing: Creative coding… & Processing A Programming handbook…) -
[7] Ethernet-kirjaston esimerkki +
[8] http://community.pachube.com/node/13 Keskustelupalsta/ohjeistus & Arduino kirjastoja +
[9] Reitittimeni valmistajan sivuilta löytynyttä tietoa, keskustelupalstojen tietoa. -
[10] Alkeellinen virheenetsintä, debug-toiminto. Puutteellinen ohjeistus, keskustelupalstoilta ei apua, eikä vastausta viestiini. -
[11] Toisen käyttäjän jakama esimerkki, arduino-keskustelupalstalta. +
[12] Oma muutos +
[13] http://arduino.cc/en/Guide/ArduinoXbeeShield helppo, toimiva esimerkki +

[14] http://www.arduino.cc/en/Main/ArduinoXbeeShield , Making Things Talk – kirja ja muita lähteitä. +
[15] Arduinon sivuilta en löytänyt tietoa. - Asia selvisi lopulta http://www.robotshop.com/content/PDF/dfrobot-arduino-shields-manual.pdf#17 +
[16] Useita lähteitä, Arduinon dokumentointi sekä muutamien verkkokauppojen tuotekuvaukset. +
[17] Useita hakuja. Pääasiassa Arduino Playground, sekä keskustelufoorumit. -

LIITE 3. KONENÄKÖ PROTOTYYPPI

1. konenäkö prototyyppi-dokumentointi 1
1.1. Tarvetieto 1
1.2. Webkameran ja Processing kirjastojen asennus 1
1.3. Kirkkaimman pisteen seuraaminen 2
1.4. Ruudulle piirtäminen 2
1.5. Hiiren kursorin vaihtaminen kirkkaimman pisteen seurantaan piirtojäljen kohdistimeksi. . 2
1.6. Videokuvan ja piirtojäljen yhdistäminen: 3
2. Käytetyt resurssit 3
2.1. Elektroniikka 3
2.2. Ratkaisutieto 3
2.3. Ohjelmat 4
2.4. Koodit 4
2.4.1. Videosyötteen lukemiseen tarvittu koodi Processing ohjelmointikielellä 4
2.4.2. ”piirtopöytä ”tason koodi Processing ohjelmointikielellä 5

1. KONENÄKÖ PROTOTYYPPI-DOKUMENTOINTI

Tämän prototyypin tavoitteena on luoda uudenlainen, visuaaliseen havaintoon perustuva käyttöliittymä, ”Köyhän miehen piirtopöytä ”. Tarkoitus on tehdä sovellus joka olisi lähes kaikkien käytettävissä ja joka rakentuisi kotoa löytyvistä tarpeista.

1.1. Tarvetieto

Sain idean tähän prototyyppiin törmätessäni Processing ohjelmointikielelle tehtyihin konenäkökirjastoihin. Ottaessani selvää kirjastojen mahdollisuuksista[1] totesin että voi olla mahdollista tehdä näköön perustuva käyttöliittymä.

Mitä vaatimuksia minulla on prototyypille?

- Mahdollisen yksinkertaiset tarpeet.
- Pisteiden seuraaminen webkameran kuvan avulla.
- Osoittimen liikuttelu ja vaikuttaminen piirtoalueeseen, esim. piirtojälki.

1.2. Webkameran ja Processing kirjastojen asennus

Webkamera on asennettu. Ei mitään ongelmia.
Kokeilen yksinkertaista processing esimerkkiä webkameran kuvan lukemiseksi, jostain syystä en saa esimerkkiä toimimaan[2].

Pienen selvittelyn jälkeen selviää että Windows:lle tarvitaan WinVDIG(QuickTime-compatible video digitizer) muuntaja jotta processingin webcam-kirjastot toimisivat[3]. WinVDIG on lakkautettu jo vuonna 2007[4]. Muutaman tunnin etsintä tuottaa tulosta, löydän halutun version 1.0.1 WinVDIG samanlaisen tilanteen kokeneen käyttäjän verkkosivustolta. Kirjoitan aiheesta viestin Processing yhteisön keskustelupalstalle ja liitän mukaan linkin WinVDIG:n lataussivulle[5].

Testaan konenäkö (engl. Computer vision) kirjastoja ja niiden ominaisuuksia selvittääkseni parhaan tavan seurata kohteita kameran avulla. En löydä tehdyistä projekteista mitään selkeää ylivoimaista ehdokasta.

1.3. Kirkkaimman pisteen seuraaminen

Sain Processing kehitysympäristöön sisällytetyn VIDEO(Capture) kirjaston BRIGHTNESS TRACKING esimerkin toimimaan. Asensin myös varmuuden vuoksi käyttäjäyhteisön tekemän Jmyron-videokirjaston, joka ei käytä WinVDIG:ä.

Kykenen nyt seuraamaan kirkkainta pistettä ruudulla.

1.4. Ruudulle piirtäminen

Hiiren osoittimella tapahtuva piirtäminen(viiva) onnistuu helposti. Processing:n ContinuousLines esimerkki toimii sellaisenaan[6].

1.5. Hiiren kursorin vaihtaminen kirkkaimman pisteen seurantaan piirtojäljen kohdistimeksi.

Tämä vaihe osoittautuu ongelmalliseksi. Processingilla on oma funktio pmouseX/Y joka vertaa kursorin sijaintia edelliseen esitettyyn ruutuun (frame). Kirkkaimman pisteen sijainti saadaan selville, mutta piirtääkseni linjan kahden pisteen välille tarvitsen kahden pisteen tiedot. Osittaisen ratkaisun löydän lisäämällä keypressed/mousepressed funktion suoritettavaksi erikseen kaikille piirtokerroille. Tämä ei ole hyvä ratkaisu sillä se hidastaa ohjelman ajoa

tuskallisesti[7]. Lopullinen ratkaisu oli yksinkertaisesti lukea kirkkaimpia arvoja tietojonoon(array) ja toteuttaa niitä järjestyksessä[8].

1.6. Videokuvan ja piirtojäljen yhdistäminen:

Yritän yhdistää yksinkertaista piirtotoimintoa videosyötteeseen. Tämä onnistuu mutta ongelmaksi muodostuu Processing:n ruudunpäivitys joka hävittää jokaisella uudella video-framella piirtojäljen alleen. Tämä johtuu siitä että sekä piirtojälki että videokuva päivittyvät samaan ikkunaan päällekkäin, jolloin vanha piirtojälki jää uuden videoruudun alle. Processing ei sisällä valmiiksi tukea erilaisille päällekkäisille tasoille (Layers). Yritän etsiä lisää tietoa ja mahdollisia ratkaisuita.

Löydän uuden Layers kirjaston[9], jonka avulla toiminnon pitäisi periaatteessa onnistua. Yhdistän videokuvan esityksen ja kirkkaimman pisteen seurannan. Jaan koodin kahteen osaan ja muokkaan tapahtumaketjua niin että video ja piirtojälki tulostuvat omille tasoilleen. Kokeilu onnistuu.
C-näppäin puhdistaa ruudun ja D-näppäin pohjassa, piirtojälkeä syntyy kameran lukemaan kirkkaimpaan pisteeseen[10].

Lopullisen prototyypin viimeistelen valitsemalla piirtoalustaksi mustan kankaan ja "kynäksi" mustaksi värjätyn kynän jossa on valkoinen kärki. Piirtopöydän tarkkuus on yllättävän hyvä.

Jatkokehitysideoita:

- videon jakaminen, yhteinen piirtotyökalu.
- fyysinen käyttöliittymä, kirkkauden seuraamisen sijasta infrapunapiste?

2. KÄYTETYT RESURSSIT

2.1. Elektroniikka

1 x Webkamera

2.2. Ratkaisutieto

Lähde	Ratkaisutieto	Epäonnistunut
Kirjat	++	
Oma tieto/IP	++	-
Keskustelupalstat, blogit, Käyttäjien tekemä	++	

dokumentointi		
tuotteiden ja yritysten dokumentointi		
Avoimen lähdekoodin tuote. Dokumentointi	+++	- -

2.3. Ohjelmat

Processing

2.4. Koodit

2.4.1. Videosyötteen lukemiseen tarvittu koodi Processing ohjelmointikielellä

```
/*

Jani Toivonen

Alkuperäiset lähteet:
Processing Layers-library
Michael Krumpus
http://nootropicdesign.com/processing-layers/index.html

Video-capture, Brightness Tracking:
Golan Levin.
*/


import com.nootropic.processing.layers.*;
import processing.video.*;
PAppletLayers layers;
Capture video;

void setup() {
  size(640, 480);
  video = new Capture(this, width, height, 30);
  smooth();
  layers = new PAppletLayers(this);
  MyLayer m = new MyLayer(this);
  layers.addLayer(m);
  cursor(CROSS);
        }

void paint() {
  if (layers != null) {
```

```
    layers.paint(this);
  } else {
    super.paint();
         }
          }

void draw() {
  if (video.available()) {
    video.read();
    image(video, 0 , 0, width, height);
                    }

          }
```

2.4.2. ”piirtopöytä ” tason koodi Processing ohjelmointikielellä

```
int num = 2;
int[] ax = new int[num];
int[] by = new int[num];
int indexPosition = 0;


class MyLayer extends PLayer {

  MyLayer(PApplet parent) {
    super(parent);
                    }

  void draw() {
    int brightestX = 0;
    int brightestY = 0;

    float brightestValue = 0;
    video.loadPixels();
    int index = 0;
    for (int y = 0; y < video.height; y++) {
      for (int x = 0; x < video.width; x++) {

        int pixelValue = video.pixels[index];

        float pixelBrightness = brightness(pixelValue);

        if (pixelBrightness > brightestValue) {
```

```
        brightestValue = pixelBrightness;
        brightestY = y;
        brightestX = x;
         ax[indexPosition] = brightestX;
         by[indexPosition] = brightestY;


indexPosition = (indexPosition + 1) % num;
      }
      index++;
    }
  }


  if(keyPressed) {
  if (key == 'd' || key == 'D') {
   //Piirtää jälkeä
    for (int i = 0; i < num; i++) {
  int pos = (indexPosition + i) % num;

  strokeWeight(50);
  stroke(250,5,5);
  line(ax[indexPosition], by[indexPosition], ax[pos], by[pos]);
                          }
                        }


   if (key == 'c' || key == 'C') {
   //Tyhjentää piirtoalueen
   background(0,0);
             }
          }
 }


}
```

[1] http://processing.org/reference/libraries/ +
[2] Tutoriaalissa tai kirjaston dokumentoinnissa ei ole mainittu ongelmasta mitään. -
[3] http://processing.org/reference/libraries/video/ +
[4] Kehotetaan käyttämään tuotetta jota ei ole ollut vuosiin? Kehittäjät jäljessä. -
[5] Yhteisö jakaa tietojaan +
[6] Processing:n mukana tullut esimerkki. +

[7] Oma ratkaisu + -
[8] Oma ratkaisu +, array esimerkki processing: Creative coding and computational art – kirjasta.
[9] http://nootropicdesign.com/processing-layers/ +
[10] Oma ratkaisu, koodien yhdistelmä uudella tavalla +

LIITE 4. POSTIHÄLYTIN PROTOTYYPPI

1. postihälytin prototyyppi-dokumentointi 1
1.1. Tarvetieto 1
1.2. Postin havaitseminen 2
1.3. etäohjaus 2
1.4. kokoonpano? 2
1.5. Sensorin toiminta 3
1.6. Testaus käytännössä 5
1.7. uusi yritys 7
2. Käytetyt resurssit 8
2.1. Elektroniikka 8
2.2. Ratkaisutieto 8
2.3. Ohjelmat 9
2.4. Koodit 9
2.4.1. Kallistuskytkimellä ja Xbee radiolla varustetun Arduinon ohjauskoodi Arduino ohjelmointikielellä 9
2.5. Sensorin tiedon vastaanottavan Arduinon ohjauskoodi Arduino ohjelmointikielellä 10

1. POSTIHÄLYTIN PROTOTYYPPI-DOKUMENTOINTI

Eräs pieni asia joka minua vaivaa, on vaihtelevat postinjakoaikataulut. Jostain syystä meille jaetaan posti joka päivä noin kello 10 - 17. Jos odotan jotain tiettyä kirjettä tai lähetystä, en voi ikinä tietää kuinka monta kertaa joudun ravaamaan laatikolla tarkistamassa tilannetta. Halusin keksiä tilanteeseen ratkaisun. Eli haluan tietää onko laatikolla käyty, ilman että joudun menemään paikan päälle tarkistamaan asiaa.

1.1. Tarvetieto

Tarvitsen sensorin joka kertoo että postilaatikko on avattu tai siellä on postia. Seuraavaksi tarvitsen tavan välittää laatikolta sisälle/johonkin josta voin tiedon nähdä.

Mitä vaatimuksia minulla on prototyypille?

- Havaita postin tuleminen
- Tiedon siirtyminen postilaatikolta näkyvälle paikalle tai sisälle
- Tilan nollaaminen ja mahd. pois/päällä toiminto

1.2. Postin havaitseminen

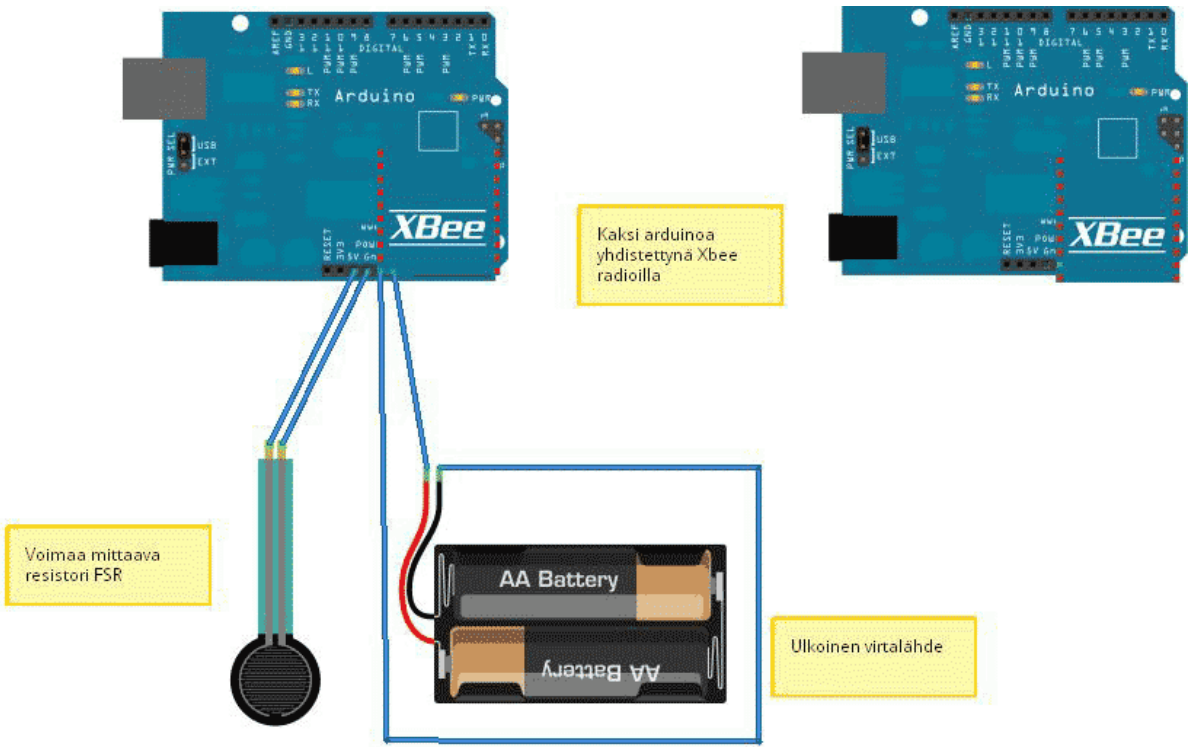
On varmasti monen monta tapaa havaita postilaatikon täyttyminen, mutta yksi ratkaisu jolla saattaa olla jatkokehitysmahdollisuuksia tulee heti mieleeni. Mikäli saisin asennettua voimaa/painoa mittaavan sensorin postilaatikon pohjalle, tietäisin että postin määrä on muuttunut.
Jatkokehitysmahdollisuutena voisi olla tapa esittää postin määrä painon mukaan, vaikkapa porrastettuna esityksenä.

1.3. etäohjaus

Päädyn etäohjauksessa käyttämään Xbee radioita, sillä minulla on jo luullakseni tarvittavat välineet ja hieman kokemusta aiemmista prototyypeistä. En tarvitse paljoa taustatietoa aloittaakseni radioiden asentamista.

1.4. kokoonpano?

Suunnittelen prototyypin kokoonpanon seuraavanlaiseksi (kuva 1).



Kuva 1. Hahmotelma prototyypistä. Toteutettu Fritzing ohjelmalla.

1.5. Sensorin toiminta

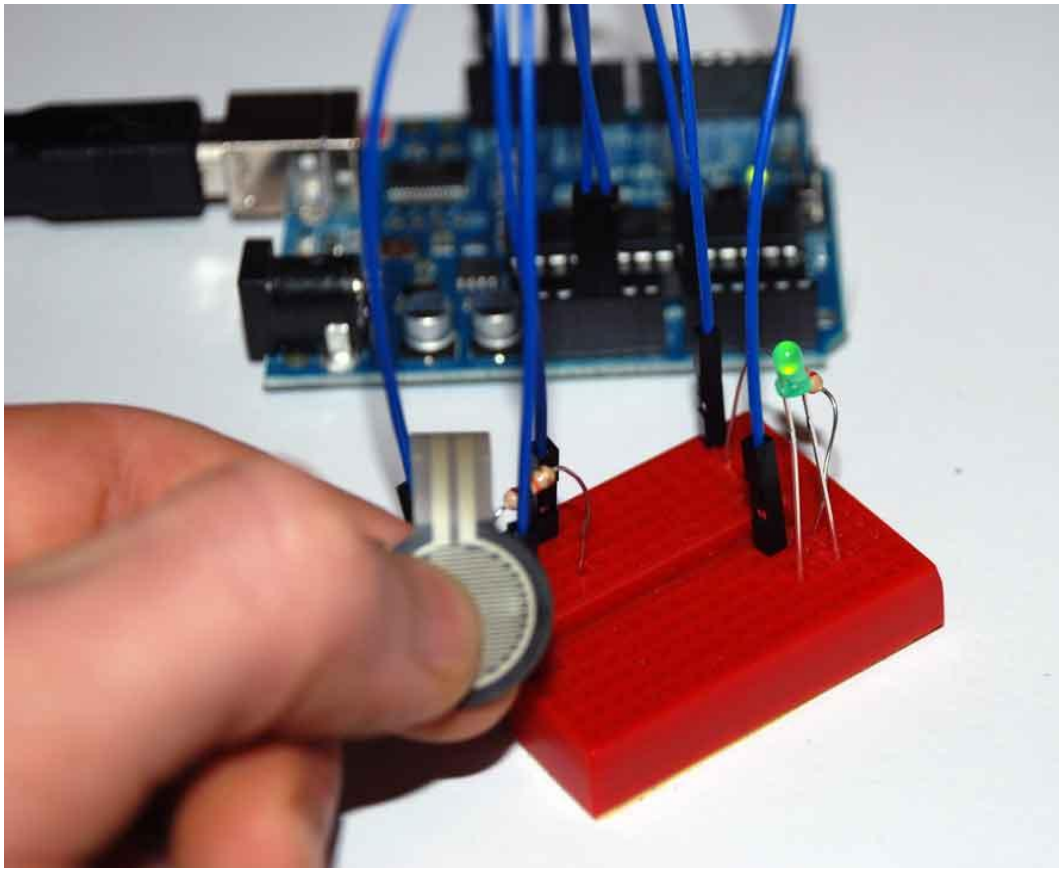
Valitsen sensoriksi yksinkertaisen ja halvan voimaa mittaavan resistorin (kuva 2).

kuva 2. Voimaa mittaava resistori (engl. Force Sensing Resistor, FSR).

Seuraavaksi testaan ja selvitä sensorin käyttöä. Verkkokauppa josta tuotteen tilasin, tarjoaa tuotedokumentoinnin. Dokumentointi on kattava ja selviääkin että pienin valmistajan lupaama mitattava ”voima ”on 100g[1]. Tämä saattaa olla ongelma, mutta päätän silti yrittää. Valmistajan tai verkkokaupan sivuilta en löydä apua sensorin testaamiseen Arduinon kanssa.

Arduinon sivustolta en jostain syystä löydä hakutoiminnolla mitään esimerkkejä FSR sensoreista, mutta sivuja selaamalla kylläkin[2]. Esimerkki on käyttäjän tekemä ja tuntuu yksinkertaiselta ja päätän kokeilla sitä.

Sensorin testaus onnistuu[3], resistorin vastus pienenee voiman lisääntyessä ja tämä antaa virran kulkea lähtöön, tässä tapauksessa LED:iin (kuva 3).

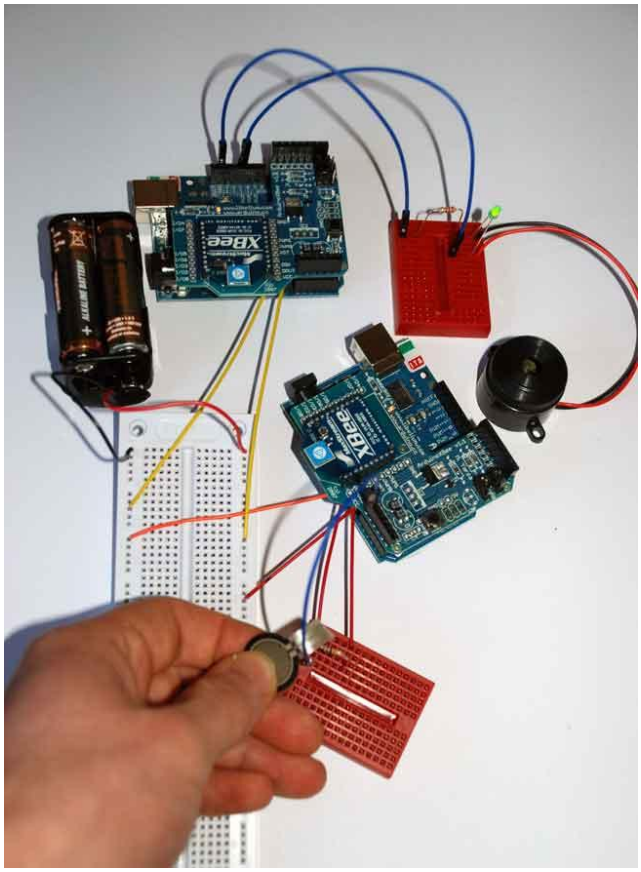


kuva 3. FSR testi onnistuu.

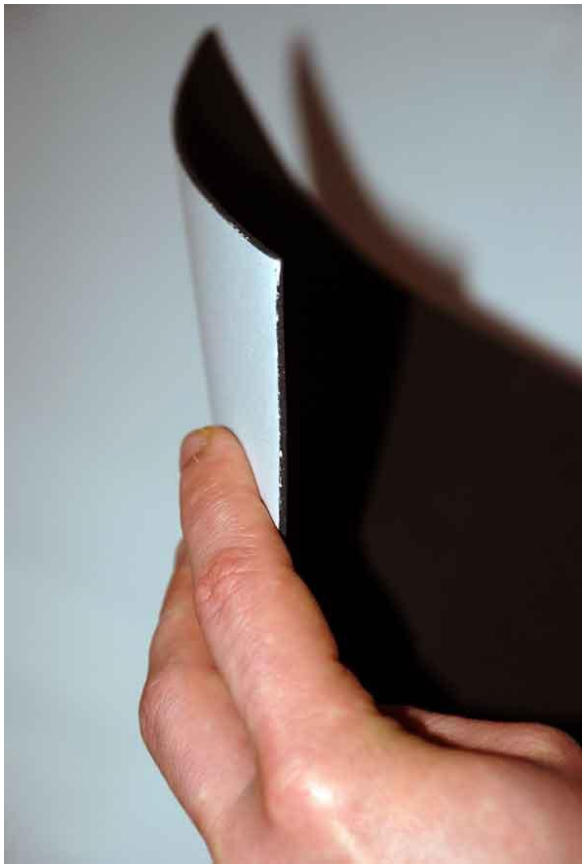
Seuraavaksi yritän välittää sensorin tiedon Xbee radioiden ja Arduinojen välillä. Käytän tähän aiemmin käyttämääni (kotivahti-prototyyppi liite 3) xbee koodin ja FSR sensorin lukemiseen tarkoitetun koodin yhdistelmää[4]. Kokeilu ei onnistu suoraan, vaan lähtönä toimiva LED palaa jatkuvasti. Arduinon Serial-monitorin avulla selvitän että sensorilla varustettu Arduino todella lähettää muuttuvaa tietoa ja että vastaanottava Arduino saa sen vastaan, sekä toteuttaa sen. Ongelma on luultavasti lähetetyn tiedon muodossa.

Selvitin asiaa ja löysin nopeasti ratkaisun Arduinon keskustelupalstalta. Ongelmana on Xbeen lähettämän tiedon muoto, ASCII tai BYTE. Yhdenmukaistamalla lähetyksen ja vastaanoton sain laitteet toimimaan halutulla tavalla[5]. Muokkasin vielä hieman koodia jotta resistorin muuttuvan tiedon tarkistus tapahtuu nopeammalla syklillä ja lähtö toimii vasta tietyn kynnyksen ylitettyään.

Tein testin jossa molemman Arduinot toimivat ulkoisella virtalähteellä ja lisäsin lähtöihin pienen piezo-summerin (kuva4). Selvitin Arduinon ulkoisen virtalähteen käyttöä, mikä selvisi helposti Arduinon dokumentoinnista[6].



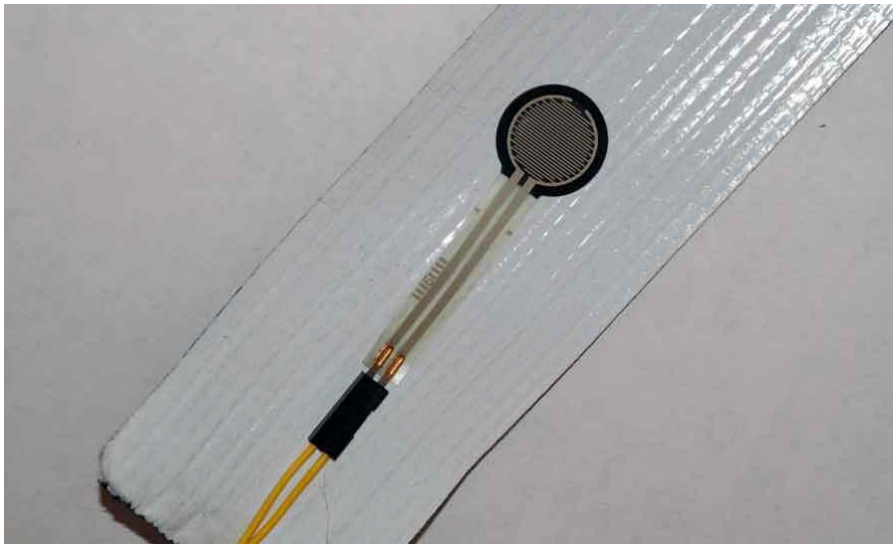
kuva 4. Testi ulkoisella virtalähteellä.



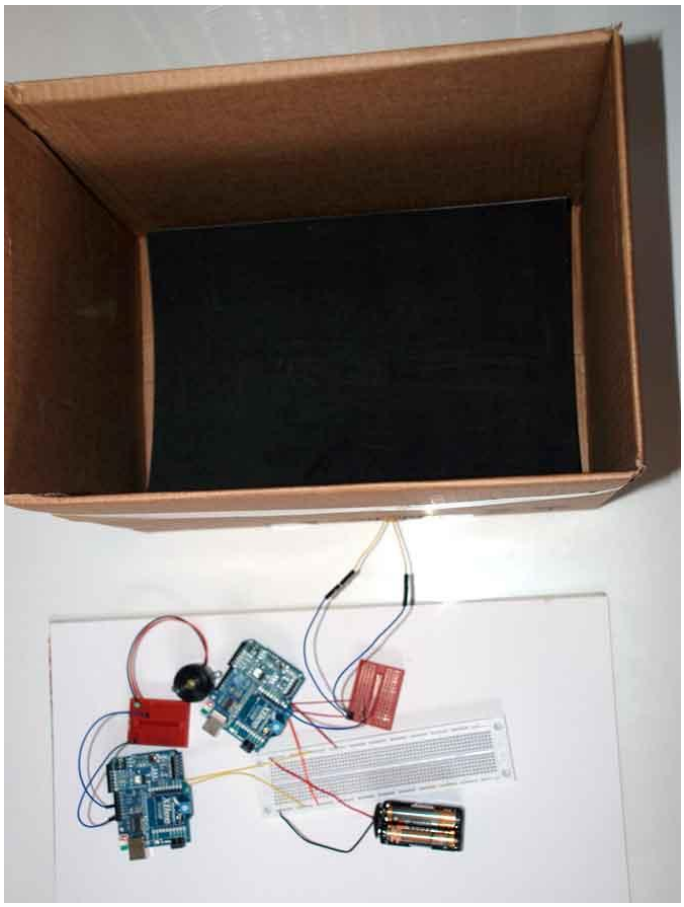
kuva 5. Ohut solumuovi

1.6. Testaus käytännössä.

Seuraava haasteeni on testata prototyypin toimintaa käytännössä. Koska protoilu vesisateessa ei kuulosta kovin järkevältä, yritän järjestää ”postilaatikkomaiset ”olosuhteet sisätiloihin. Postilaatikoksi käy pieni pahvilaatikko. Löydän käyttäjän tekemän projektin jossa vastaavia FSR sensoreita on käytetty yhdessä ohuen solumuovimaton kanssa. Muovi on tarpeeksi jäykkää välittääkseen iskuja, mutta antaa samalla suojaa hennolle elementille[7] (kuva 5). FSR sensorin teippasin solumuovin (kuva6) pohjaan ja asetan pahvilaatikon pohjalle (kuva 7). Olen valmis aloittamaan postin pudotuskokeet.



kuva 4. FSR teipissä.



kuva 5. ”postilaatikko ”valmiina testiin.

Testin tulokset ovat heikot[8]. kirjekuoret tai ohuet sanomalehdet eivät riitä laukaisemaan sensoria. Vaihtoehtoina olisi lisätä sensorien määrää tai vaihtaa solumuovi johonkin kovempaan pintaan. Vaistoni sanoo, ettei kumpikaan niistä toisi tyydyttäviä tuloksia. Joudun miettimään jotain uutta ratkaisua.

1.7. uusi yritys

Etsin tietoa projekteista joissa on käytetty jotain sensoria oven tai luukun avaamisen tai vaihtoehtoisesti painon/liikkeen havaitsemiseen.
Muutaman vesiperän jälkeen löydän ratkaisun, joka tuntuu liian helpolta ollakseen totta. Ratkaisu löytyy käyttäjän projektista[9].

Kallistuskytkin (kuva 6) kytkeytyy päälle tietyn asennon tai kallistuskulman myötä. Osa on lähes ilmainen ja helppokäyttöinen.

kuva 6. yksinkertainen kallistuskytkin.

Löydän kytkimelle esimerkin Arduinon sivuilta[10]. Muokkaan esimerkkiä niin että se sopisi projektiini. Yritän sulauttaa olemassa olevaa ja uutta koodia. Useista yrityksistä huolimatta en onnistu käynnistämään Lähtöä toisessa Arduinossa[11]. Joudun palaamaan lähtöruutuun ja testaamaan kytkintä yksinään, ilman Xbee radioita ja kahta Arduinoa.

Onnistun testissä yhden Arduinon kanssa. Kirjoitan osan koodista uusiksi niin että se toimii kahden Arduinon ja radioiden kanssa. Testi toimii[12]. Lisään vastaanottavaan päähän vielä On/off kytkimen. Tämä mahdollistaa vastaanottavan laitteen tilan nollaamisen.

Mitä saavutettiin?

- Onnistuin rakentamaan järjestelmän jossa sensori havaitsee liikkeen (postilaatikon kannen avaamisen) ja viestittää tiedon langattomasti.
- Vastaanottava pää toteuttaa ohjelmoinnin mukaisesti käskyn ja käynnistää lähdöt.

Jatkokehitysideoita:

- Useiden laitteiden lisääminen ja monen Xbeen käyttö matriisiverkossa. Fyysinen hallintapaneeli jota voisi laajentaa modulaarisesti tarpeen mukaan.
- Projektin osien hiominen ja molempien kokoonpanojen rakentaminen piirilevyille, sekä laatikointi lopulliseen muotoon.
- Verkkovirran käyttö vastaanottavassa päässä. 9-12V.

2. KÄYTETYT RESURSSIT

2.1. Elektroniikka

2 x Arduino Duemilanove
2 x Xbee shield
2 x Xbee series 1 radio
1 x FSR sensori
1 x Kallistuskytkin
1 x 10 KOhm resistori
Koekytkentälevyjä
Hyppykaapeleita
Serial –USB kaapeleita
Ulkoisia virtalähteitä

Erilaisia lähtöjä:
Ledejä
Piezo summeri

2.2.Ratkaisutieto

Lähde	Ratkaisutieto	Epäonnistunut
Kirjat, e-kirjat, maksulliset oppaat.		
Oma tieto/IP	2	2
Keskustelupalstat, blogit ja Käyttäjien tekemä dokumentointi.	4	
Kaupallisten, suljetun lähdekoodin tuotteiden dokumentointi ja yritysten jakama tieto.	1	
Avoimen lähdekoodin tuotteiden dokumentointi ja avoimen lähdekoodin yhteisöjen ”viralliset ” esimerkit.	2	1

2.3. Ohjelmat

Processing
Arduino
Fritzing

2.4. Koodit

2.4.1. Kallistuskytkimellä ja Xbee radiolla varustetun Arduinon ohjauskoodi Arduino ohjelmointikielellä

```
/*
Alkuperäinen koodi:
 created 2005
 by DojoDave <http://www.0j0.org>
 modified 17 Jun 2009
 by Tom Igoe
  http://www.arduino.cc/en/Tutorial/Button

  Muokattu 9.4.2010
  Jani Toivonen
 */


const int buttonPin = 2;
int buttonState = 0;

void setup() {
    Serial.begin(9600);
  pinMode(buttonPin, INPUT);
}

void loop(){
  buttonState = digitalRead(buttonPin);
  if (buttonState == HIGH) {
Serial.print('H');
  }
  if (buttonState == LOW) {
  Serial.print('L');

  }

    delay(100);
}
```

2.5. Sensorin tiedon vastaanottavan Arduinon ohjauskoodi Arduino ohjelmointikielellä

```
/*
Koodi kirjoitettu 9.4.2010
Jani Toivonen
Alkuperäinen lähde:
http://arduino.cc/en/Guide/ArduinoXbeeShield
*/
const int buttonPin = 2;
const int ledPin =  12;

int incomingByte;
void setup() {
  Serial.begin(9600);
  pinMode(ledPin, OUTPUT);
}

void loop() {
  if (Serial.available() > 0) {
    incomingByte = Serial.read();
      Serial.println(incomingByte);
    if (incomingByte == 'H') {
      digitalWrite(ledPin, HIGH);
    }
    if (incomingByte == 'L') {
      digitalWrite(ledPin, LOW);

    }

    }

    delay(100);

    }
```

[1] Valmistajan dokumentointi +
[2] Hakutoiminnon puutteellisuus -
[3] Käyttäjän tekemä esimerkki +
[4] Oma ratkaisu +
[5] Arduinon Keskustelupalsta +
[6] Arduinon dokumentointi +
[7] Käyttäjän tekemä projekti +

[8] Oma epäonnistuminen, tiedon puute -
[9] Käyttäjän tekemä projekti +
[10] Virallinen esimerkki, Arduino +
[11] Oma tieto, yritys epäonnistuu -
[12] Oma tieto, muokkaus +